ANALISA DAN IMPLEMENTASI METODE KOHONEN *NEURAL NETWORK* UNTUK PENGENALAN KARAKTER HURUF ARAB

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

WELI ANDRIAN
10751000241

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2012

ANALISA DAN IMPLEMENTASI METODE KOHONEN *NEURAL NETWORK* UNTUK PENGENALAN KARAKTER HURUF ARAB

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

WELI ANDRIAN
10751000241

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2012

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul **"ANALISA DAN IMPLEMENTASI METODE KOHONEN NEURAL NETWORK UNTUK PENGENALAN KARAKTER HURUF ARAB"**. Tugas akhir ini dilakukan sebagai syarat untuk memperoleh gelar sarjana teknik di Jurusan Teknik Informatika UIN SUSKA Riau dengan beban kredit sebanyak 6 SKS.

Pada kesempatan ini penulis juga hendak mengucapkan terima kasih kepada berbagai pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini :

1. Prof. Dr. H. M. Nazir, selaku Rektor Universitas Islam Negeri Sultan Syarif Kasim Riau.
2. Dra. Yenita Morena, M.Si, selaku Dekan Fakultas Sains dan TeknologiUniversitas Islam Negeri Sultan Syarif Kasim Riau.
3. Novriyanto, ST, M.Sc, selaku Ketua Jurusan Teknik Informatika, FakultasSains dan Teknologi.
4. Iwan Iskandar, M.T, sebagai koordinator tugas akhir yang telah banyak membantu dan mempersiapkan kebutuhan dalam proses administrasi tugas akhir.
5. Ibu Lestari Handayani, S.T, M.Kom, sebagai pembimbing penulis yang banyak memberikan saran dan masukan yang banyak membantu dalam proses penyelesaian tugas akhir.
6. Ibu Luh Kesuma Wardhani, M.T, sebagai penguji I.
7. Ibu Siska Kurnia Gusti, S.T, sebagai penguji II.
8. Ibu dan Ayah penulis yang selalu mengingatkan dan memberikan dukungan penuh kepada penulis, semoga allah selalu memberikan berkahnya didunia maupun diakhirat kepada kalian.

9. Abang dan kakak penulis yang selalu memberikan motivasinya.
10. Kepada adik-adik penulis yang ada dikampus Irma Welly, Erzi Hidayat, Rahmayuni Iskandar, Andrian Hadinata, Yandiko, Nanda, Salam, Aina dan lainnya.
11. Pizaini, Ari, Hendra, Ismail, Detha, Inggih, Ligar, Teguh, Satria, Pio, bang Faisal, dan teman-teman seperjuangan lainnya khususnya TIF kelas B 2007.
12. Kepada rekan-rekan seperjuangan yang tergabung dalam tim Pekanbaru Animation dan Riau Citrasoft.
13. Semua pihak yang terlibat baik secara langsung ataupun tidak langsung dalam pelaksanaan tugas akhir ini yang tidak dapat penulis sebutkan satu persatu.

Penulis berharap dapat mendapatkan nilai yang baik dalam pelaksanaan Tugas Akhir ini. Penulis menyadari masih terdapat banyak kekurangan. Oleh sebab itu, penulis mengharapkan masukan berupa kritikan dan saran yang bersifat membangun. Semoga tugas akhir ini bermanfaat bagi pembaca dan dunia penelitian kampus.

Pekanbaru, Desember 2012

Penulis

ANALISA DAN IMPLEMENTASI METODE KOHONEN *NEURAL NETWORK* UNTUK PENGENALAN KARAKTER HURUF ARAB

WELI ANDRIAN
10751000241

Tanggal Sidang : 27 Desember 2012
Periode Wisuda : Februari 2013

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Huruf Arab merupakan suatu huruf yang penting bagi umat Islam dan dianjurkan untuk dipelajari dikarenakan digunakan pada Al-quran dan hadist. Huruf Arab sendiri digunakan oleh sekitar 234 juta orang yang tersebar pada berbagai negara Arab, Persia, Urdu dan lainnya. Pengenalan pola merupakan suatu ilmu dari bidang studi jaringan syaraf tiruan untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu objek. Pengenalan karakter yang merupakan bagian bidang studi pengenalan pola yang banyak dipelajari dan telah menghasilkan banyak aplikasi yang sangat berguna. Kohonen *neural network* atau dikenal juga dengan *self organizing map* (SOM)merupakan suatu metode yang banyak digunakan untuk pengenalan suatu pencitraan yang diperkenalkan pertama kali oleh Tuevo Kohonen dan telah banyak digunakan untuk pengenalan berbagai karakter seperti karakter huruf Latin, Hiragana, Kanji serta Arab. Namun, studi untuk pengenalan karakter huruf Arab yang jauh lebih sedikit jumlahnya dibanding karakter lainnya menjadi hal yang melatar belakangi pentingnya dilakukannya penelitian ini. Pengujian dari hasil pengenalan karakter menggunakan Kohonen ini dilakukan dengan menggunakan rumus matematis (jumlah sampel berhasil / jumlah sampel uji) x 100%. Hasil pengujian memperlihatkan tingkat persentase mencapai 80% (*very good*) untuk 15 sampel pola dari beberapa karakter *isolated*, 77,14% (*satisfactory*) untuk 140 sampel pola karakter dari semua karakter *isolated* dan 70,32% (*satisfactory*) untuk 310 sampel pola karakter dari semua karakter *isolated, beginning* dan beberapa dari karakter *middle* dan *end.*

Kata Kunci : Karakter Arab, Kohonen *neural network*, Pengenalan karakter, Pengenalan pola, SOM.

ANALYSIS AND IMPLEMENTATIONOF THE KOHONEN NEURAL NETWORK METHOD FORARABIC CHARACTER RECOGNITION LETTERS

WELI ANDRIAN
10751000241

Date of Final Exam : December 27, 2012
Graduation Ceremony Period : February 2013

Informatics Department
Facultyof Sciences and Technology
State IslamicUniversity of Sultan Syarif Kasim Riau

ABSTRACT

Arabic alphabet is an important alphabetic for Islamic people and recommended to be studied due to the Arabic was used in the Al-quran and hadist. The Arabic alphabet it self was used around 234 million peoples that were spread on several Arabic nations, Persia, Urdu and other. Pattern recognition is a studies from neural network discipline that used to classify or describe something based on quantitative measurement features (characteristic)or the main properties of an object. Character recognition that is the studies of pattern recognition has been studied and has been produced a lot of very useful applications. Kohonen neural network, also know as self organizing map (SOM), is a widely used method for recognize of the imaging that was introduce by Tuevo Kohonen and has been widely used for the recognition of various characters such as Latin characters, Hiragana, Kanji, and Arabic. However, the study of Arabic characters that far fewer than the research of the other characters is had been the important background of this research. The testing was done using a mathematical formula (number of succesful samples/number of test samples) x 100%. The results of the testing showed that the recognition rate was reached 80% (very good) for 15 pattern character samples from a fews from isolated characters, 77,14% (satisfactory) for 140 pattern character samples from all of isolated characters and 70,32% (satisfactory) for 310 patterns character samples from all of isolated, beginning and several from middle, and end characters.

Keywords **:** *Arabic character, Character recognition, Kohonenneural network, Pattern recognition,* SOM.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ALGORITMA

DAFTAR RUMUS

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Huruf Arab atau yang disebut juga dengan huruf hijaiyah sangat berhubungan erat dengan Islam dan dianjurkan untuk dipelajari di karenakan huruf dan bahasa Arab merupakan suatu huruf yang digunakan pada Al-quran dan Hadits. Huruf Arab sendiri digunakan oleh sekitar 234 juta orang yang banyak penggunanya adalah dari beberapa suku Arab yang tersebar di berbagai negara yang mayoritas Arab, Persia, serta Urdu. Untuk versi standar huruf Arab yang banyak digunakan pada negara-negara Arab tersebut disebut juga dengan *Modern Standard Arabic* (MSA) yang merupakan sebuah versi standar yang digunakan untuk komunikasi resmi yang melintasi berbagai negara Arab (Lorigo, 2006).

Pattern recognition (pengenalan pola) merupakan disiplin ilmu yang membahas mengenai cara-cara pengklasifikasian suatu objek ke beberapa kelas atau kategori dan mengenali kecenderungan suatu data (Santosa, 2007). Dalam perkembangannya, objek-objek yang diteliti pada bidang ilmu ini digunakan untuk beragam aplikasi yang banyak manfaatnya seperti pengenalan pola wajah, pengenalan pola kartu kredit, *image* atau *signal*, pola tanda tanganatau pengukuran lain yang perlu diklasifikasikan atau dicari fungsi regresinya.

Character recognition (pengenalan karakter) merupakan bagian dari bidang studi *pattern recognition* (pengenalan pola) yang banyak dipelajari dan telah menghasilkan banyak aplikasi yang sangat berguna. *Pattern recognition* (pengenalan pola) merupakan suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu objek (Putra, 2010). Penelitian mengenai *character recognition* telah menghasilkan proses pada pengenalan pola terhadap beberapa karakter huruf seperti pengenalan karakter huruf latin, tulisan tangan, huruf hiragana, huruf kanji, huruf Arab serta huruf lainnya.

Ide dasar yang ingin dicapai dari proses pengenalan karakter ini adalah untuk mengekstrak pola dari suatu *input* dan menerapkan suatu algoritma untuk mengklasifikasikan pola karakter kepada kelas yang sesuai (Peyarajan, 2011).

Artificial Neural network (ANN) atau jaringan syaraf tiruan merupakan suatu cabang ilmu yang diinspirasi dari sistem jaringan syaraf makhluk hidup. ANN digunakan sebagai alternatif pendekatan yang konvensional yang biasanya kurang fleksibel terhadap perubahan suatu struktur masalah. Ada beberapa karakteristik kemampuan otak manusia seperti mengingat, menghitung, mengeneralisasi, adaptasi, serta konsumsi energi yang rendah. Diharapkan ANN dapat melakukan dan meniru cara kerja otak manusia seperti kemampuan-kemampuan di atas. ANN berusaha meniru struktur / arsitektur dan cara kerja otak manusia sehingga mampu menggantikan beberapa pekerjaan manusia. Pekerjaan seperti mengenali pola (*pattern recognition*), prediksi klasifikasi, pendekatan fungsi, optimisasi adalah pekerjaan-pekerjaan yang diharapkan bisa diselesaikan dengan ANN (Santosa, 2007).

Metode pembelajaran dari *Pattern recognition* (pengenalan pola) dibagi menjadi 2, yaitu *supervised* (terbimbing)dan *unsupervised* (tidak terbimbing). *Supervised* yaitu apabila vektor fitur pelatihan telah tersedia dan telah diketahui kelas-kelasnya. Yang termasuk kedalam pembelajaran *unsupervised* adalah LVQ (*Learning Vector Quantization*), *Backpropagation, Perceptron,* Hopfield dan lain-lain. Sedangkan *unsupervised* adalah apabila vektor fitur pelatihan tidak tersedia sebelumnya sehingga untuk proses klasifikasi, sekumpulan vektor fitur dikelompokkan kedalam beberapa *cluster* (*group*) berdasarkan tingkat kemiripannya. *Unsupervised* disebut juga dengan *clustering* (Putra, 2010).

Kohonen *neural network* atau dikenal juga dengan *self organizing map* merupakan suatu metode yang banyak digunakan untuk pengenalan suatu pencitraan yang diperkenalkan pertama kali oleh Tuevo Kohonen. Didalam suatu proses *character recognition* yang mempunyai basis metode kohonen tidak terdapat *hidden layer* seperti pada kebanyakan metode *character recognition* lain yang memungkinkan jalannya proses pengenalan karakter menjadi lebih cepat

dibandingkan dengan metode *supervised* yang memiliki *hidden layer*. Proses pengenalan dari metode ini meggunakan fungsi ketetanggaan untuk menentukan *property* dari topologi *input*-nya(Shatil, 2006).

Penelitian pengenalan karakter menggunakan metode Kohonen telah banyak diimplementasikan oleh peneliti sebelumnya seperti pada karakter huruf latin, karakter huruf *Bangla* (India), hiragana (Jepang) dan lainnya. Penelitian-penelitian sebelumnya menunjukkan bahwa Kohonen *neural network* dapat mengenali bermacam-macam karakter-karakter yang ada di berbagai negara tersebut. Namun, dibanding penelitian karakter yang lain, menurut Mezghani (2006) penelitian mengenai pengenalan karakter huruf Arab dikatakan lebih sedikit dan terbatas. Ini tidak lain di karenakan bentuk dari karakter huruf Arab yang lebih rumit dibanding karakter huruf lainnya.

Dari beberapa penjelasan di atas didapat suatu kebutuhan akan penelitian pengenalan karakter huruf Arab sebagai studi kasusdi karenakan penelitian pengenalan karakter huruf Arab sendiri memiliki tingkat kerumitan dan pola yang berbeda dibandingkan dari huruf-huruf lainnya. Oleh karena itu, penulis menetapkan suatu konsep penelitian untuk menganalisa dan mengimplementasikan metode kohonen *neural network* untuk pengenalan karakter huruf Arab.

1.2. Rumusan Masalah

Berdasarkan dari latar belakang yang telah dijelaskan, maka rumusan masalah penelitian ini adalah bagaimana menganalisa dan mengimplementasikan Metode Kohonen *Neural Network* untuk Pengenalan Karakter dengan studi kasus karakter huruf Arab.

1.3. Batasan Masalah

1. Aplikasi yang dibuat adalah aplikasi yang melakukan proses pengenalan karakter, bukan suku kata atau pun kata.

2. Pengenalan karakter mengenali bentuk huruf Arab terpisah (*isolated*), di awal kalimat (*beginning*), di tengah kalimat (*middle*), dan di akhir kalimat (*end*).
3. Matriks yang digunakan berukuran 10 x 10 *pixel*.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah melakukan analisa dan implementasi terhadap metode Kohonen *Neural Network* untuk pengenalan karakterhuruf Arab.

1.5. Sistematika Penulisan

Berikut merupakan rencana susunan sistematika penulisan laporan Tugas Akhir yang akan dibuat :

BAB I Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan sistematika penulisan dari Tugas Akhir yang dibuat.

BAB II Landasan Teori

Bab ini membahas tentang teori-teori umum dan khusus yang berhubungan dengan tugas akhir ini.

BAB III Metodologi Penelitian

Bab ini membahas langkah-langkah yang dilaksanakan dalam proses penelitian, yaitu pengamatan pendahuluan, tahapan identifikasi masalah, perumusan masalah, analisa aplikasi, perancangan aplikasi dan implementasi beserta pengujian.

BAB IV Analisa dan Perancangan

Bab ini berisi pembahasan mengenai kebutuhan sistem, yang terdiri dari : *Flowchart system, data collection, pre-processing, processing* serta

classification yang menggunakan algoritma dari metode kohonen *neural network*.

BAB V Implementasi dan Pengujian

Bab ini berisi penjelasan mengenai implementasi yang terdiri dari: batasan implementasi, lingkungan implementasi, hasil implementasi, pengujian sistem dan kesimpulan pengujian.

BAB VI Kesimpulan dan Saran

Bagian ini berisi kesimpulan yang dihasilkan dari pembahasan tentang analisa dan implementasi metode Kohonen *neural network* untuk pengenalan karakter huruf Arab.

BAB II
LANDASAN TEORI

2.1. *Artificial Neural Netwok* (ANN)

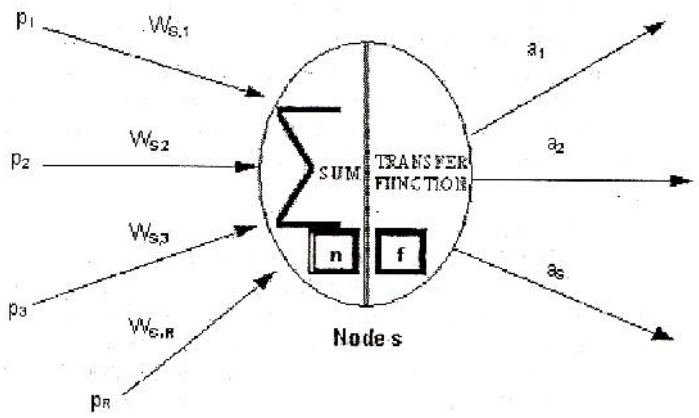
Jaringan syaraf tiruan atau *Artificial Neural Network* (ANN) merupakan suatu model jaringan *neural* (syaraf) yang arsitekturnya meniru prinsip kerja dari syaraf otak manusia. Model pertama ANN pertama kali muncul pada tahun 1943 yang diperkenalkan oleh McCulloch dan Pitts. Model tersebut dibuat berdasarkan fungsi *neuron* biologis yang merupakan dasar unit sinyal dari sistem syaraf (Putra, 2010).

ANN memiliki kemampuan seperti yang dimiliki oleh otak manusia, seperti :

1. Kemampuan untuk belajar dari pengalaman.
2. Kemampuan untuk melakukan perumpamaan (*generalization*), terhadap *input* baru dari pengalaman yang dimilikinya.
3. Kemampuan memisahkan (*abstraction*) karakteristik penting dari *input* yang mengandung data yang tidak penting.

2.1.1. Model *Artificial Neural Network*

Proses pemodelan ANN menggunakan pendekatan pemodelan *black box.* Cara kerjanya didasari pada mekanisme kerja penyaluran informasi sistem *neural network.* Namun, dikarenakan keterbatasan yang dimiliki oleh struktur ANN maka hanya sebagian kecil dari kemampuan sistem syaraf manusia yang dapat ditiru (Putra, 2010).



Gambar 2.1. Ilustrasi Model ANN (Putra, 2010)

p_r menyatakan sinyal *input* dari *node input* ke i = 1, 2, …, R, dengan R menyatakan jumlah *input.*

$W_{S,r}$ menyatakan bobot (*weight*) hubungan dari *node* (*neuron*) *input* r ke *node* (*neuron*) yang dituju j, j = 1, 2, …, S, dengan S menyatakan jumlah *neuron.*

n menyatakan total (jumlah) sinyal terbobot yang masuk ke *node* s atau juga sering disebut sebagai tingkat pengaktifan (*activation level*) di *node* s.

f menyatakan fungsi transfer (*transfer function*) yang akan menentukan *output* dari *node* s dan tergantung pada nilai n.

a_s menyatakan sinyal yang keluar (*outgoing signal*) atau *output* dari *node* s.

Berikut adalah tabel perbandingan antara sistem *neural network* (NN) dan sistem *artificial neural network* (ANN). Dalam tabel berikut dapat dilihat prinsip kerja NNyang ditiru oleh ANN.

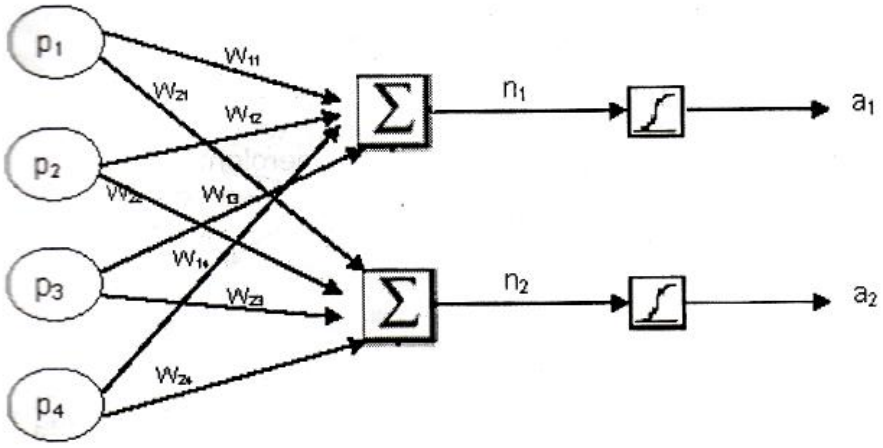
Tabel 2.1. Spesifikasi *Neural Network* (NN) dan *Artificial Neural Network* (ANN)

Neural Network	*Artificial Neural Network*
Soma (sel tubuh)	*Node*
Dendrit	Sinyal *input*
Sinyal pada akson	Sinyal *output*
Synapsis	Bobot
Memiliki kecepatan rendah	Memiliki kecepatan tinggi

Memiliki *neuron* (syaraf) sekitar 100 miliar	Hanya memiliki sekitar ratusan *neuron* (syaraf)

Secara ringkas, prinsip kerja *neuron* dapat dinyatakan sebagai berikut. Pada suatu *neuron*, sinyal input yang diterima oleh dendrit akan masuk ke *node* (soma). Pada *node* terjadi proses penjumlahan sinyal *input* yang telah terbobot (dinyatakan dengan $W_{s,r}\,P_r$) dan dilambangkan dengan n. Penjumlahan sinyal terbobot tersebut (n) diproses menjadi menjadi sinyal *output* (a) dengan mengunakan suatu fungsi aktivasi. Sinyal *output* ini kemudian diteruskan ke *neuron* lain oleh akson.

Berikut adalah struktur ANN yang memiliki 4 *input*, 2 *neuron*, dan 2 *output.*



Gambar 2.2. Contoh ANN dengan 4 *input,* 2 *neuron,* dan 2 *output* (Putra, 2010)

2.1.2. Prosedur *Artificial Neural NetworkLearning*

Proses *learning* yang disebut juga dengan pembelajaran atau pelatihan (*training*) pada ANN merupakan suatu proses perubahan atau penyesuaian tingkat kekuatan hubungan antara *node* yang saling terhubung. Tingkat kekuatan hubungan antar *node* dinyatakan dalam nilai bobot. Dengan kata lain, proses *learning* dalam ANN mrupakan proses penyesuaian nilai-nilai bobot tersebut (Putra, 2010).

Selama proses *learning*, faktor bobot akan mengalami perubahan dan bila tahapan *learning* sudah selesai maka nilai faktor bobot yang dihasilkan akan

disimpan dan digunakan sebagai faktor bobot terpakai. Fungsi dari faktor bobot pada ANN adalah untuk memperkuat (*amplifying*), menghambat (*attenuating*), dan mengubah (*change*) sinyal yang masuk.

Ada beberapa algoritma yang biasa digunakan dalam proses *learning* yang terbagi kedalam proses *supervised learning* (pembelajaran terbimbing) dan *unsupervised learning* (pembelajaran tak terbimbing).

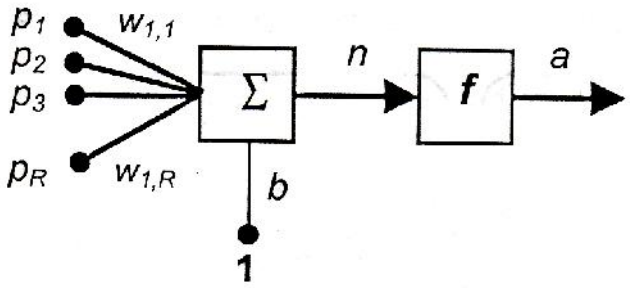
2.1.2.1.*Supervised Learning* (Pembelajaran Terbimbing)

Supervised learning tidak melibatkan kompetisi, namun menggunakan agen eksternal (*output* aktual) untuk setiap pola *input* yang membimbing proses *learning.* Ada beberapa bentuk dari *supervised learning.* Bentuk yang paling sederhana, *force learning,* bekerja sebagai berikut : pada saat sebuah *input* pola (sebuah *input* vektor) di presentasikan, sebuah *neuron* yang tepat dipaksa untuk melakukan tindakan dari lingkungan luar. *Neuron* aktif yang terhubung akan berkembang pada kekuatan koneksi dari waktu ke waktu, respon dari *neuron* akan menjadi lebih sensitif pada pola *input* dan belajar mengklasifikasikan dengan benar tanpa dorongan dari lingkungan luar. Bentuk lain dari *force learning* adalah *reinforcement learning,* pada jaringan yang menerima umpan balik baik *output* positif atau negatif dan menggunakan informasi ini untuk meningkatkan respon dari waktu ke waktu. Dua metode ini mengimplikasikan Hebbian *learning* (2006, Samarasinghe).

Bentuk ketiga yang lebih baik, bentuk dari *supervised learning* yang berkembang dari metode di atas adalah *error correction learning* (pembelajaran koreksi kesalahan), yang sekarang umumnya dikenal sebagai *supervised learning,* yang nilai aktual dari *output* yang benar ditampilkan ke jaringan dan bobotnya disesuaikan sampai perbedaan yang sebenarnya antara *output* dari respon *neuron* dan *output* yang benar diterima. Gagasan ini lebih kompleks dibandingkan dengan Hebbian *learning,* dan telah dikembangkan menjadi metode pembelajaran yang lebih kuat berdasarkan gradien *error.* Berikut adalah beberapa metode yang termasuk kedalam *supervised learning.*

2.1.2.1.1.*Perceptron*

Perceptron merupakan algoritma jaringan syaraf tiruan yang paling sederhana yang diperkenalkan pada tahun 1957 oleh Rosenblatt. *Perceptron* digunakan untuk melakukan klasifikasi terhadap pola-pola yang terpisah secara linier (*linearly separability*) (Putra, 2010). Berikut adalah gambar dari arsitektur *perceptron* dengan *neuron* tunggal.



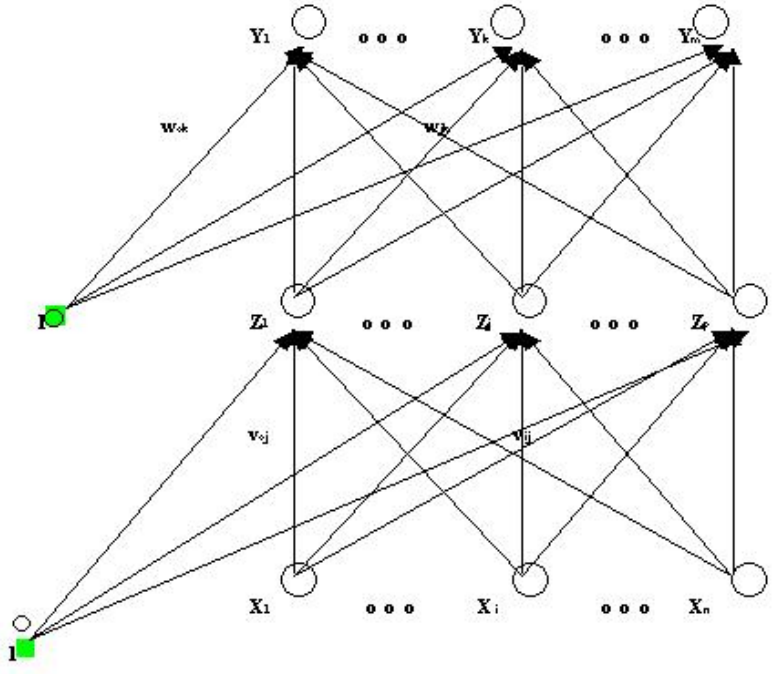
Gambar 2.3. Arsitektur *Perceptron* Dengan Neuron Tunggal (Putra, 2010)

2.1.2.1.2.*Backpropagation*

Backpropagation adalah salah satu metode untuk *training multilayer neural network* yang meggunakan *learning rule gradient descend* (Santosa, 2007). *Backpropagation* diperkenalkan pertama kali oleh Rumelhart, Hinton dan William pada tahun 1986, lalu dikembangkan lagi pada tahun 1988 oleh Rumelhart dan Mc Clelland. *Backpropagation* merupakan jaringan yang menggunakan aturan *feedforward* (umpan maju). Tahapan *training* (pelatihan) *backpropagation* meliputi tiga tahap (Subiyanto, 2010) :

1. Masukkan *input* secara *feedforward* (umpan maju)
2. Hitung dan propagasi balikkan kesalahan yang bersangkutan
3. Atur bobot koneksi

Berikut adalah gambar yang mengilustrasikan arsitektur jaringan *backpropagation* dengan *multilayer* dengan satu *hidden layer* (unit Z), unit *output* (Y) dan unit *hidden layer* (Z) mempunyai bias. Bias pada *unit output* tipikal Y_k dinotasikan dengan w_{ok}, bias pada unit *output* tipikal Z_j dinotasikan dengan v_{oj}. Tindakan bias ini seperti pada bobot koneksi yang unit biasnya selalu mengeluarkan nilai 1.



Gambar 2.4. Arsitektur *Backpropagation* (Subiyanto, 2010)

2.1.2.1.3.*Learning Vector Quantization* (LVQ)

Jaringan LVQ merupakan algoritma jaringan syaraf tiruan yang diperkenalkan oleh Tuevo Kohonen. LVQ merupakan salah satu jaringan syaraf tiruan yang melakukan proses *learning* secara *supervised* (terbimbing). LVQ mengklasifikasikan *input* secara berkelompok kedalam kelas yang sudah didefinisikan melalui jaringan yang telah dilatih.dengan kata lain LVQ mendapatkan n *input* mengelompokkan

kedalam m *output.* Arsitektur dari jaringan LVQ ini terdiri dari *input, layer* kohonen, dan *layeroutput.*

2.1.2.2.*Unsupervised Learning* (Pembelajaran Tak Terbimbing)

Unsupervised Learning atau pembelajaran tak terbimbing mempelajari bagaimana suatu sistem dapat belajar untuk menghasilkan pola *input* tertentu dengan cara merefleksikan struktur statistik dari keseluruhan koleksi pola *input.* Sebaliknya dengan *supervised learning* (pembelajaran terbimbing), tidak ada taget *output* atau evaluasi lingkungan yang berhubungan dengan tiap *input* (Dayan, 1999).

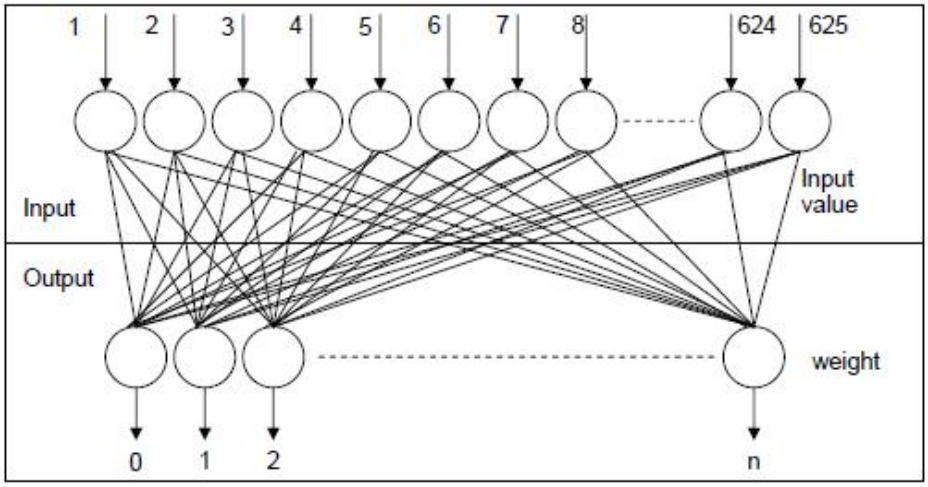
Unsupervised learning merupakan suatu metode yang penting dikarenakan lebih banyak keberadaannya pada syaraf dibanding *supervised learning. Unsupervised learning* secara umum memiliki sebuah sejarah panjang dan berbeda-beda. Beberapa penelitian awal dilakukan oleh Horace Barlow pada tahun 1992 yang mendapatkan cara dari karkteristik kode *neural,* Donald MakKay pada tahun 1956, mengadopsi sebuah teori *cybernetic-theoretic,* dan David Marr pada tahun 1970 yang membuat sebuah penelitian awal dalil *unsupervised learning* mengenai tujuan dari pembelajaran pada model dari *neocortex.* Hebb *rule* pada 1949, yang berhubungan secara statistik ke penelitian *neurophysiological* pada elastisitas, juga memiliki kontribusi besar.

Clustering memberikan contoh yang cukup baik, misalkan suatu kasus yang *input*-nya adalah kegiatan fotoreseptor yang diciptakan oleh berbagai gambar apel atau jeruk. Pada ruang lingkup dari segala aktifitas yang memungkinkan, *input-input* tertentu membentuk dua kelompok, dengan derajat sedikit banyak dari variasi 10^6, yaitu dimensi yang lebih rendah. Salah satu tugas alami untuk pembelajaran *unsupervised learning* adalah untuk mencari dan mengkarakteristikkan *cluster* yang terpisah dan berdimensi rendah ini.

2.2. *Kohonen Neural Network*

Kohonen *Neural network* adalah sebuah metode *neural network* yang dikenalkan oleh Tuevo Kohonen yang mempunyai hanya 2 *layerneuron*, *layer input dan layer output.* Kohonen *neural network* memiliki perbedaan arsitektur dari backpropagation *neural network*. Perbedaan pertama, kohonen *neural network* tidak memiliki *hidden layer.* Yang kedua, kohonen memiliki cara pemanggilan *pattern* dan prosedur pembelajaran yang berbeda. Ketiga, kohonen tidak menggunakan fungsi aktivasi, dan terakhir kohonen tidak menggunakan bobot yang tidak seimbang pada *neural network* (Shatil, 2006).

Berikut adalah ilustrasi dari *layer* kohonen *neural network :*

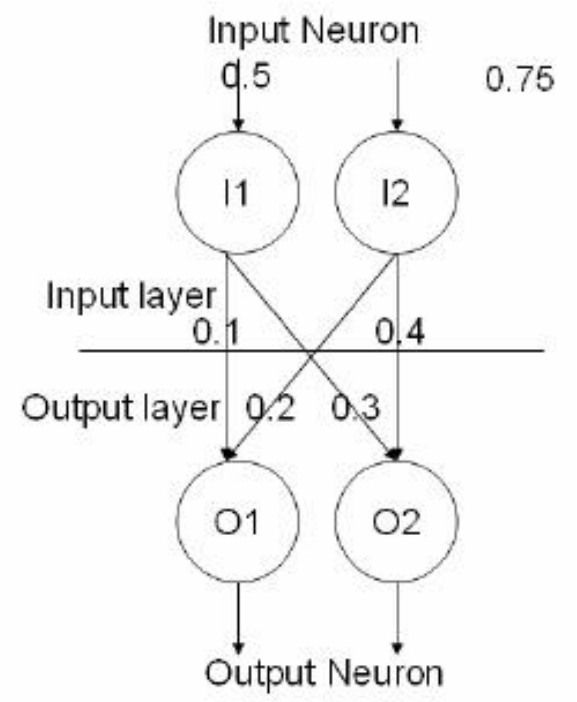


Gambar 2.5. Kohonen *neural network* (Shatil, 2006)

2.2.1. Struktur Kohonen *Neural Network*

Kohonen *neural network* hanya berisi *layer input* dan *layer output*, tidak ada *hidden layer.Input* ke kohonen *neural network* menggunakan *input neuron*. *Input neuron* masing-masing diberi nomor *floating point* yang membentuk *input pattern* ke jaringan *neural network*. Sebuah kohonen *neural network* membutuhkan *input* yang dinormalisasikan dengan kisaran antara -1 dan 1. Penyajian sebuah *input pattern* ke *network* akan menyebabkan reaksi dari *outputneuron*. *Output* dari kohonen *neural*

network sangat berbeda dari *output* metode *neural network* lainnya seperti backpropagation *neural network*. Pada backpropagation, jika kita memiliki *neural network* dengan lima *output neuron* kita akan diberi output yang terdiri dari lima nilai. Namun, dalam sebuah kohonen *neural network* hanya satu dari *output neuron* yang benar-benar menghasilkan nilai tunggal, baik benar atau salah. Ketika*pattern* ini disajikan kedalam kohonen*neural network*, satu *output neuron* tunggal dipilih sebagai *neuron output*(Heaton, 2005).



Gambar 2.6. Struktur Kohonen *Neural Network* (Shatil, 2006)

2.2.2. Algoritma Kohonen *Neural Network*

Menurut Laurene Fausett proses dari kohonen *neural network* dapat dilakukan oleh algoritma sebagai berikut (Fausett, 1993) :

a. Langkah 0 : Inisialisasi bobot $w_{ij,}$ radius tetangga, dan parameter *learning rate* (α)

b. Langkah 1 : Saat kondisi stop bernilai salah, maka lakukan langkah 2-8

c. Langkah 2 : Untuk tiap vektor *input* x, lakukan langkah 3-5

d. Langkah 3 : Untuk tiap j, lakukan perhitungan seperti berikut :

$$D(j) = \sum_i (w_{ij} - x_i)^2 \quad \text{.....(2.1)}$$

e. Langkah 4 : Temukan *index* J yang memiliki nilai D(J) terkecil

f. Langkah 5 : Update semua bobot untuk tiap unit j dengan rumus sebagai berikut :

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})] \quad \text{....(2.2)}$$

g. Langkah 6 : *Update learning rate* (α) dengan mengalikan *learning rate* yang lama dengan 0,5

h. Langkah 7 : Kurangi radius tetangga pada waktu yang spesifik

i. Langkah 8 : Cek kondisi *stop*

2.3. *Character Recognition* (Pengenalan Karakter)

Semua tipe pengenalan karakter memiliki proses *recognition* (pengenalan), ada beberapa langkah yang dilakukan untuk sebuah pengenalan karakter. Berikut adalah prosedur dari pengenalan karakter (Shatil, 2006) :

Pertama, yang diperlukan adalah sejumlah besar data mentah atau data yang dikumpulkan yang akan diproses dan untuk kemudian akan dilakukan proses *training* oleh sistem. Lalu, data yang telah dikumpulkan dilakukan proses perbandingan dengan sejumlah data yang mirip. Selanjutnya yang harus diperhatikan adalah level kompleksitas dari data yang telah dikumpulkan, karena tahapan berikutnya akan sangat bergantung pada tipe data yang telah dikumpulkan tersebut. Tipe data itu seperti suatu dokumen hasil pemindaian (*scan*) atau dokumen hasil tulisan tangan.

Kedua adalah tahapan *pre-processing*. Pada tahapan inilah sebagian besar prosedur pemrosesan *image* diolah. Seperti konversi *image* kedalam tipe *grayscale*,

konversi kedalam format biner, koreksi kemiringan image. Hasil tahapan *processing* tergantung pada tahapan *pre-processing.* Jadi, desain dari tahapan *pre-processing* harus diperhatikan dengan baik.

Ada beberapa kategori metode algoritma yang digunakan untuk tahapan *pre-processing* diantaranya metode berbasis *histogram shape, clustering, entropy, object attribute* dan *spatial.* Kategori yang sesuai dengan tahapan proses *pre-processing* adalah algoritma berbasis *clustering* yang menggunakan sampel *grayscale image*, berikut adalah beberapa dari algoritma yang berbasis *clustering* (Sankur, 2004) :

1. Algoritma Otsu

   Pada tahun 1979, Otsu mengenalkan sebuah algoritma yang menggunakan analisa *shape* (bentuk). Algoritma ini merupakan algoritma pencarian biner yang paling terkenal (Athimethphat, 2011). Algoritma Otsu melakukan pengecilan terhadap jumlah bobot diantara *class* dari *foreground* dan *background pixel* untuk membangun sebuah tahapan permulaan / *preprocessing* yang optimal. Algoritma ini memberikan hasil yang memuaskan ketika jumlah *pixel* pada tiap *class* dekat satu dengan yang lain. Pilihan yang optimum diformulasikan sebagai berikut (Sankur, 2004):

   $$T_{opt} = \arg\max[P(T).(1-P(T)).(m_f(T)-m_b(T))^2] \quad \text{..... (2.3)}$$

   Algoritma Otsu memiliki kelemahan apabila kualitas *image* buram atau kabur serta *background image* yang kompleks algoritma ini sering gagal dalam menjalankan prosesnya (Athimethphat, 2011).

2. Algoritma Lloyd

   Algoritma ini mengasumsikan suatu *image* dapat dikarakteristikkan oleh sebuah campuran distribusi dari *pixel foreground* dan *background* dari suatu *image* $p(g) = P(T).p_f(g) + (1-p(T)).p_b(g)$. Dibawah asumsi dari fungsi densitas varian Gaussian yang sama, ambang batas yang meminimalkan kesalahan keseluruhan menjadi :

$$T_{opt} = \arg\min[\frac{m_f(T)+m_b(T)}{2} + \frac{\pm^2}{m_f(T)-m_b(T)}\log\frac{1-P(T)}{P(T)}] \quad \text{..... (2.4)}$$

Hasil minimum yang menghasilkan ambang optimal dari ekspresi diatas dapat ditemukan melalui sebuah pencarian iterasi.

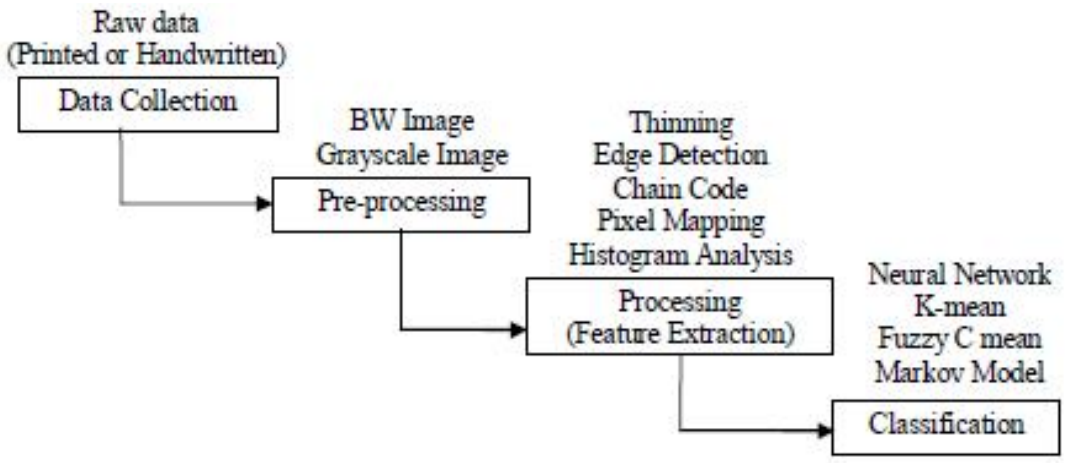
3. Algoritma Riddler

   Algoritma ini adalah salah satu skema iterasi pertama yang berdasarkan pada dua *class* Gaussian *mixture model.* Pada iterasi n, sebuah ambang baru T_n dibentuk dengan menggunakan rata-rata *class* dari *foreground* dan *background* seperti berikut :

$$T_{opt} = \lim_{n\to\infty} T_n \quad \text{dimana} \quad T_{n+1} = \frac{m_f(T_n)+m_b(T_n)}{2} \quad \text{..... (2.5)}$$

Yang ketiga, tahapan *processing* yang terdiri dari *thinning* (penipisan), *edge detection, chain code, pixel mapping,* serta *histogram analysis* adalah beberapa proses dari tahapan *processing.* Tahapan ini pada dasarnya melakukan proses konversi data mentah menjadi komponen yang dapat diolah melalui proses *training.*

Terakhir, setelah data dilakukan proses *training* dan pengenalan (*recognition*) data akan diproses pada tahapan *classification.* Data yang telah diolah dari tahapan *pre-processsing* dan *processing* akan melakukan proses *training* yang berarti mengajarkan sistem tentang data yang masuk, sehingga untuk selanjutnya dapat dengan mudah mengenali sebuah data masukan.

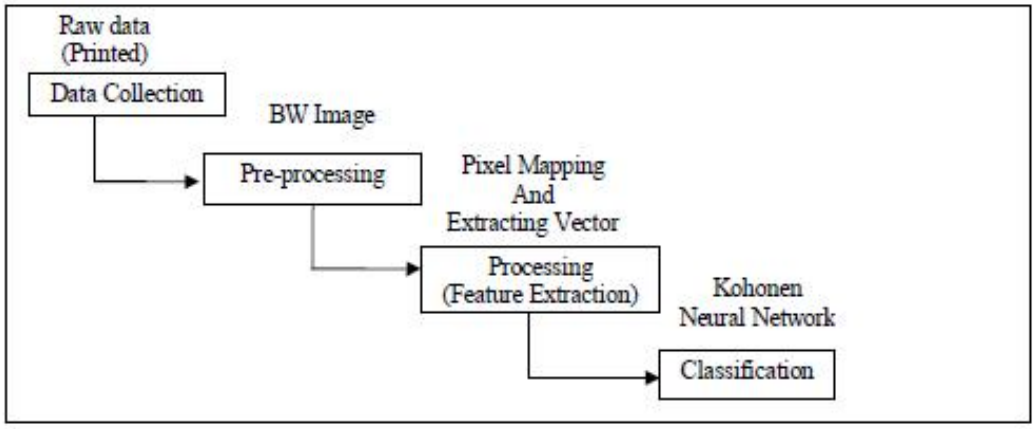


Gambar 2.7. Prosedur Pengenalan Karakter (Shatil, 2006)

2.3.3. C*haracter Recognition* dengan Kohonen *Neural Network*

Prosedur dalam pengenalan karakter menggunakan Kohonen *neural network* tidak terlalu jauh berbeda dengan prosedur pengenalan karakter secara umum. Berikut adalah tahapan dari pengenalan karakter dengan Kohonen neural network (Shatil, 2006) :

1. Pola karakter diambil dari data mentah (*row data*)
2. Karakter yang telah didapat dikonversi menjadi bentuk *grayscale* dan *image* hitam putih pada tahapan *pre-processing*
3. Jumlah *pixel* akan dipetakan kedalam suatu area spesifik lalu objek vektor akan diekstrak dari *image* yang berisikan suatu karakter. Tahapan ini disebut juga dengan tahapan *processing*
4. Tahapan terakhir adalah *classification* yang menggunakan algoritma dari Kohonen *neural network.*



Gambar 2.8. Prosedur Pengenalan Karakter dengan Kohonen Neural Network (Shatil, 2006)

2.4. *Data Collection*

Dikarenakan untuk mengembangkan sistem pengenalan karakter bergantung pada data mentah yang dikumpulkan, tipe data tersebut menjadi sesuatu yang sangat penting dalam proses pengenalan karakter. Dalam proses *data collection* ini, sebelum *image* diproses pada sistem *image* tersebut terlebih dahulu akan disesuaikan ukurannya sesuai dengan ukuran yang telah kita tetapkan, misalnya sebuah *image* yang berukuran 250 x 250 *pixel*. Pada tahapan ini tidak dilakukan koreksi kemiringan dan lainnya, sehingga ukuran dan bentuk *image* harus sangat diperhatikan.

2.5. *Pre-processing*

Berikut adalah tahapan dari proses *pre-processing* menggunakan algoritma Otsu :

2.5.1. Konversi Citra ke *Grayscale*

Untuk menjadikan suatu citra atau objek gambar menjadi *image* biner, suatu citra tersebut harus terlebih dahulu di konversi dulu kedalam format *grayscale.*

Perhitungan untuk mendapatkan nilai *grayscale* untuk suatu citra *pixel* adalah sebagai berikut :

Grayscale = R 1/30 + G 1/59 + B 1/11 ..... (2.6)

2.5.2. Konversi *Grayscale* ke *Binary Image*

Proses selanjutnya dari tahapan *pre-processing* adalah proses konversi *image grayscale* menjadi *image* biner. Sebelum mengubah bentuk *grayscale* menjadi *image* biner, dicari terlebih dahulu nilai *threshold* dari histogram nilai *grayscale.* Caranya adalah dengan mencari dari nilai histogram di nilai *grayscale* mana terjadi perubahan intensitas yang cukup signifikan (kontras), maka akan diambil nilai tengah dari nilai *grayscale* tersebut sebagai nilai *threshold.*

Setelah didapatkan nilai *threshold,* selanjutnya adalah mengubah *image* biner dengan kondisi jika nilai *grayscale* dari *pixel* dibawah nilai *threshold,* maka *pixel* menjadi *pixel* hitam. Dan jika nilai *grayscale* dari *pixel* sama dengan nilai *threshold,* maka *pixel* menjadi putih.

2.6. *Processing (Feature Extraction)*

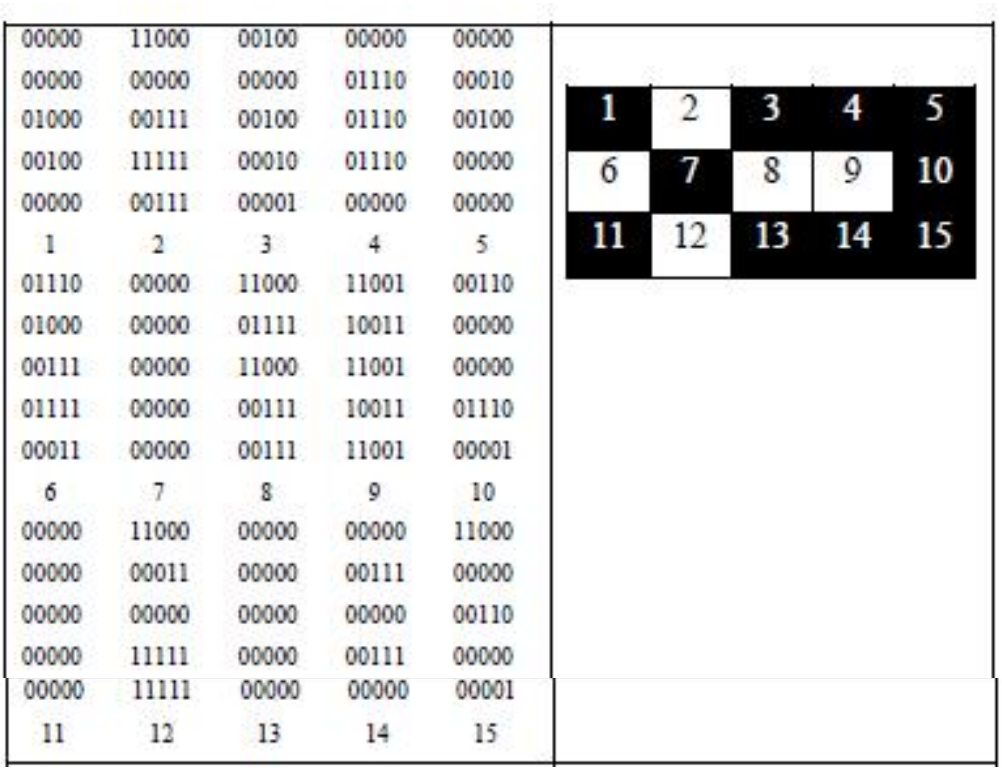
Setelah tahapan *pre-processing* selesai, tahapan selanjutnya adalah *processing* atau disebut juga dengan *feature extraction*. Tujuan utama dari *processing* ini adalah mencari sebuah vektor dalam sebuah *image. Processing* memiliki beberapa tahapan dalam mengekstrak sebuah *image* vektor. Apabila suatu *image* diproses dan dikonversi kedalam *image* biner pada tahapan sebelumnya, maka akan tersedia 2 tipe data dari *image* tersebut, yaitu 0 dan 1. Bilangan 1 untuk area yang berwarna putih dan bilangan 0 untuk area yang berwarna hitam. Untuk melakukan tahapan *processing* ini, suatu *image* perlu dikonversi menjadi vektor yang memiliki panjang 625 untuk setiap karakter. Berikut adalah langkah-langkah dari tahapan *processing* :

1. Pemetaan ke area sampel

2. *Creating Vector*

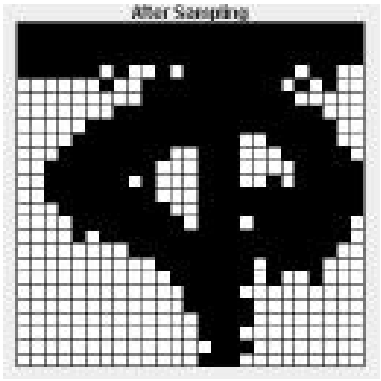
2.6.1. Pemetaan ke Area Sampel

Dari tahapan sebelumnya didapatkan bilangan biner yang telah terbagi kedalam tiap-tiap kelas *pixel.* Sekarang yang harus dilakukan adalah mencari untuk tiap-tiap 5x5 *pixel* dengan cara memberikan sebuah *unique number* untuk tiap kelasnya. *Unique number* ini merujuk kepada 5x3 area *pixel.* Selanjutnya tentukan area 5x5 *pixel* akan menjadi sebuah area yang berwarna hitam atau area yang berwarna putih. Untuk menentukannya adalah dengan mengambil prioritas dari jumlah bilangan 0 atau 1 dari area 5x5 *pixel.* Sehingga apabila bilangan 0 memiliki jumlah yang lebih banyak pada area 5x5 *pixel* lokasi i^{th} maka pada posisi i^{th} akan diberikan area berwarna hitam, begitu juga sebaliknya untuk bilangan biner 1 yang menunjukkan area berwarna putih.



Gambar 2.9. Pemetaan Area Sampel (Shatil, 2006)

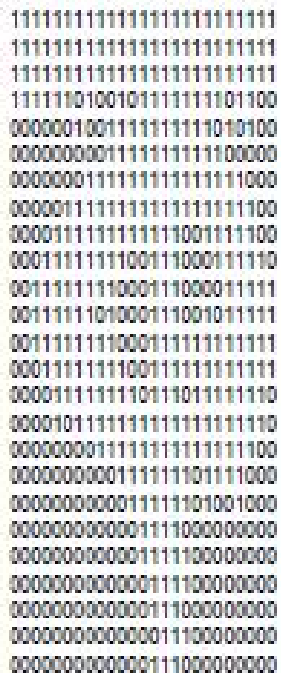
Berikut adalah contoh dari sebuah pola yang telah melalui proses perubahan ukuran menjadi 25x25 persegi area *pixel*.



Gambar 2.10. Pola 25x25 *pixel* (Shatil, 2006)

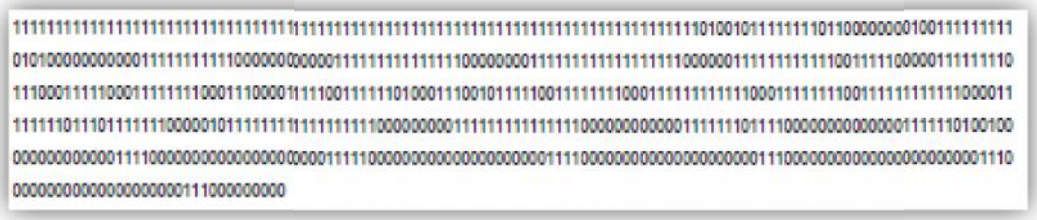
2.6.2. *Creating Vector*

Ketika telah didapat sampel dari area bilangan biner yang memiliki area hitam dan putih, selanjutnya tentukan bilangan 1 untuk persegi area hitam dan 0 untuk persegi area putih. Berikut adalah gambar dari representasi dari bilangan 0 dan 1 dari sampel pola gambar 2.7.



Gambar 2.11. Representasi Pola (Shatil, 2006)

Untuk membuat representasi sebuah vektor lakukan pengumpulan bilangan biner tiap baris, lalu gabungkan. Berikut adalah representasi dari vektor pola pada Gambar 2.8.



Gambar 2.12. Representasi Vektor

2.7. *Classification* (Kohonen *Neural network*)

Banyak proses yang telah dilakukan pada tahapan *pre-processing* dan *processing.* Tujuan dari tahapan-tahapan tersebut adalah untuk menyederhanakan serta dapat diproses oleh Kohonen *neural network.* Tahapan *classification* dari vektor

input yang telah didapat melalui proses *processing* diproses lagi dengan menggunakan algoritma dari metode Kohonen *neural network.*

2.8. *Distance* (Jarak)

Jarakadalah suatu metode yang digunakan untuk menentukan kesamaan (*similarity degree*) atau ketidaksamaan (*dissimilarity degree*) antara dua vektor fitur. Tingkat kesamaan yang dibandingkan berupa suatu nilai (*score*) dan berdasarkan nilai tersebut dua vektor fitur akan dikatakan mirip atau tidak(Putra, 2010).

Ukuran jarak harus memenuhi syarat-syarat sebagai berikut (Santosa, 2007) :

1. $d(x, y) \geq 0$ (non-negatif)
   Tidak ada jarak yang mempunyai nilai negatif.
2. d(x, y) = jika dan hanya jika x = y (*identity of indiscernibles*)
   Jarak antara suatu objek atau titik dengan objek atau titik itu sendiri adalah nol.
3. d(x, y) = d(x, y) (simetri)
   Jarak dari x ke y adalah sama dengan jarak dari y ke x.
4. $d(x, z) \leq d(x, y) + d(y, z)$ (ketidaksamaan segitiga).

Berikut ini adalah beberapa jenis dari metode jarak yang dapat digunakan untuk mengukur tingkat kemiripan dua buah vektor (Putra, 2010) :

1. *Euclidean Distance*
   Euclidean distance adalah metrika yang paling sering digunakan untuk menghitung kesamaan 2 vektor, *Euclidean distance* menghitung akar dari kuadrat perbedaan 2 vektor (*root of square differences between* 2 *vectors*).
   Berikut ~~adalah rumus~~ dari *Euclidean distance :*
   $d_{ij} = \sqrt{\sum_{k=1}^{n} (xik - xjk)^2}$ ..... (2.7)

   Berikut adalah contoh yang memiliki 2 buah vektor :

A = [0, 3, 4, 5]

B = [7, 6, 3, -1]

Euclidean distance dari vektor A dan B adalah :

$d_{ij} = \sqrt{((0-7)^2 + (3-6)^2 + (4-3)^2 + (5-(-1))^2}$

$= \sqrt{(49 + 9 + 1 + 36)} = 9.747$

Euclidean distance adalah kasus istimewa dari Minkowski *distance* dengan λ = 2.

2. *City Block Distance*

City block distance disebut juga dengan Manhattan *distance / boxcar distance / absolute value distance. City block distance* menghitung nilai perbedaan absolut dari 2 vektor (*absolute differences between 2 vectors*). Berikut adalah rumus dari *city block distance* :

$$d_{ij} = \sum_{k=1}^{n} |x_{ik} - x_{jk}|$$

..... (2.8)

Contoh :

Diberikan 2 vektor A dan B sebagai berikut :

A = [0, 3, 4, 5]

B = [7, 6, 3, -1]

Hasil perhitungan jaraknya menurut *city block distance* adalah :

d_{AB} = |0 - 7| + |3 - 6| + |4 - 3| + |5 + 1|

= 7 + 3 + 1 + 6 = 17

City block distance adalah kasus istimewa dari Minkowski *distance* dengan λ = 1.

3. *Minkowski Distance*

   Minkowski *distance* merupakan metrika dengan pola ordo λ yang menggeneralisasi beberapa metrika sebelumnya, dimana λ = 1 dinyatakan debagai *city block distance* dan λ = 2, dinyatakan dengan *Euclidean distance.* Chebishev *distance* merupakan kasus khusus dari Minkowski *distance* dengan λ = ∞ (tak terhingga).

   Berikut adalah rumus untuk menghitung jarak dari Minkowski *distance* :

   $d_{ij} = \sqrt{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^{\lambda}}$ ..... (2.9)

   Berikut contoh perhitungan jarak antara 2 vektor menggunakan Minkowski *distance* ordo 3 :

   A = [0, 3, 4, 5]

   B = [7, 6, 3, -1]

   $d_{AB} = \sqrt[3]{(0-7)^3 + (3-6)^3 + (4-3)^3 + (5-(-1))^3}$

   $= \sqrt[3]{343 + 27 + 1 + 216}$

   $= \sqrt[3]{587} = 8.373$

4. *Hamming Distance*

   Urutaan biner 0 dan 1 dinamakan *word* didalam teori *coding.* Jika dua *word* memiliki panjang sama, dapat dihitung jumlah dimana posisi *word* berbeda. Jumlah *digit* yang berlainan disebut *Hamming distance.*

   Berikut adalah rumus unutk menghitung jarak menggunakan Hamming *distance* :

   $d_{ij} = q + r$ ..... (2.10)

Keterangan :

q = jumlah dari variabel dengan nilai 1 untuk citra i dan 0 untuk citra j

r = jumlah dari variabel dengan nilai 0 untuk citra i dan 1 untuk citra j

Berikut contoh dari vektor A dan B :

A = [1, 1, 1, 1]

B = [0, 1, 0, 0]

Karena :

q = jumlah dari variabel dengan nilai 1 untuk vektor A dan 0 untuk vektor B = 3

r = jumlah dari variable dengan nilai 0 untuk vektor A dan 1 untuk vektor B = 0

Didapat 3 *digit* berbeda yang perbedaanya adalah (q = 3 dan r = 0), sehingga Hamming *distance*-nya adalah 3.

Menurut pelitian perbandingan antara beberapa metode jarak yang dilakukan oleh Syamani (2008), metode *city block / Manhattan distance* dan beberapa lainnya lebih unggul dalam proses kecepatan. Namun dalam hal akurasi *euclidean distance* sedikit lebih unggul dibanding yang lainnya, selain itu *euclidean distance* juga lebih banyak digunakan dalam berbagai penelitian.

2.9. Pengujian *Recognition* dan *Error*

Dari hasil pengenalan karakter yang dilakukan sebelumnya, dicarilah performansi keberhasilan dalam mengenali karakter. Pengujian performa pengenalan karakter dan *error* dilakukan dengan menggunakan rumus matematis yaitu (Ratih, 2009) :

(Jumlah berhasil / jumlah pengujian) x 100% ..... (2.11)

Tingkat pengenalan karakter dikatakan sangat baik apabila tingkat akurasinya mencapai lebih dari 90% *excellent* (tinggi sekali), 89%-80% very good (sangat tinggi), 79%-64% satisfactory (tinggi), 63%-51% sufficient (cukup), 50%-0% insufficient (rendah) (Koerich, 2003).

2.10. Pengujian *Black Box*

Menurut *Black* (2007), pengujian *black box* atau *behavioral* merupakan pengujian mengenai apa yang dilakukan oleh sistem, terutama perilakunya (*behavior*) dan masalah-masalah bisnis. *Black box* dilakukan untuk mengidentifikasi kesalahan (*bug*) yang terdapat pada hasil-hasil, pemrosesan dan perilaku dari sistem. Pengujian *black box* biasanya dilakukan oleh *tester*.

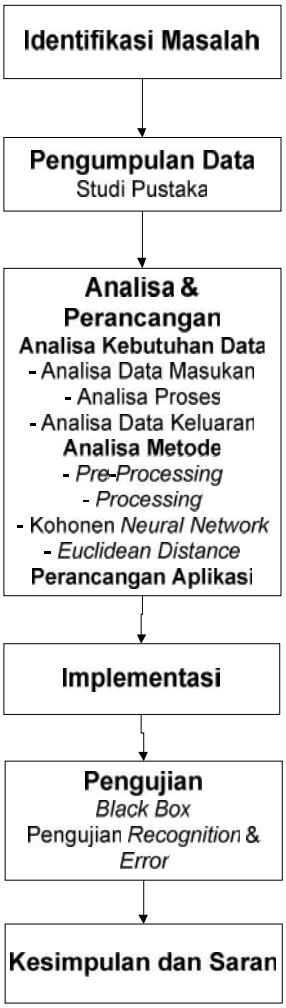
Pengujian *black box* berfokus pada perilaku eksternal dari suatu *software* atau berbagai komponennya sambil memandang objek yang diuji sebagai sebuah kotak hitam (*black box*) sehingga mencegah *tester* untuk melihat isi didalamnya. Pengujian *black box* memverifikasi penanganan yang benar dari fungsi-fungsi eksternal yang disediakan oleh *software* atau apakah perilaku yang diamati tersebut memenuhi harapan *user* atau spesifikasi produk. Bentuk paling sederhana dari pengujian *black box* adalah dengan mulai menjalankan *software* dan melakukan pengamatan dengan harapan mudah untuk membedakan hasil yang diharapkan dan mana tidak. Bentuk ini disebut juga *ad hoc testing.* (Tian, 2005).

Setelah dilakukan pengujian berulang kali dan ditentukan bahwa masalah-masalah yang terjadi karena *software* dan bukan karena *hardware*, maka informasi tersebut disampaikan kepada pihak yang bertanggungjawab untuk memperbaiki masalah-masalah tersebut. Bentuk lain dari pengujian *black box* adalah penggunaan *checklist* yang spesifik berisikan daftar-daftar fungsi eksternal apa yang seharusnya ada serta beberapa informasi mengenai kinerja yang diharapkan atau pasangan *input* dan *output*.

Menurut Tian (2005), selama *component* atau *subsystem test, tester* berfokus pada kesalahan-kesalahan yang ada pada komponen-komponen sistem. Jika komponen tersebut bisa berdiri sendiri, maka dapat digunakan teknik *black box.*

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian menjelaskan tentang tahapan-tahapan yang dilakukan pada proses penelitian agar berjalan sesuai dengan tujuan yang telah ditentukan sehingga mencapai hasil yang baik. Tahapan penelitian dapat dilihat dari gambar berikut :



Gambar 3.1. Tahapan Metodologi Penelitian

3.1. Identifikasi Masalah

3.1.1. Penelitian Pendahuluan

Tahapan penelitian pendahuluan merupakan tahapan pencarian informasi awal mengenai penelitian-penelitian yang sudah dilakukan oleh peneliti-peneliti terdahulu pada bidang pengenalan karakter secara umum dan juga pengenalan karakter huruf arab, serta metode kohonen *neural network*.

3.1.2. Identifikasi Masalah

Tahapan identifikasi masalah dilakukan setelah mendapatkan innformasi-informasi yang dibutuhkan dari penelitian pendahuluan. Tahapan ini dilakukan untuk mengetahui permasalahan yang terjadi. Dari penelitian pendahuluan diketahui bahwa dengan melihat pentingnya pengetahuan huruf arab bagi umat islam baik bagi orang dewasa maupun anak-anak, dalam hal ini penulis mengidentifikasi terdapat suatu kebutuhan aplikasi pengenalan karakter huruf arab sebagai media pembelajaran untuk anak-anak yang diimplementasikan pada aplikasi *desktop* untuk mempermudah proses belajar menulis huruf arab.

Berdasarkan dari identifikasi permasalahan yang telah dijelaskan, maka penulis merumuskan penelitian ini akan membahas mengenai analisa dan implementasi metode kohonen *neural network* untuk pengenalan karakter huruf Arab.

3.2. Pengumpulan Data

Tahapan pengumpulan data merupakan tahapan yang digunakan untuk memperoleh informasi dan data dari permasalahan yang ada. Ada beberapa tahapan dalam pengumpulan data ini, yaitu :

a. Studi Pustaka

   Studi pustaka dilakukan dengan mempelajari informasi dan data yang didapat dari buku, jurnal ilmiah serta penelitian lain yang berhubungan sebagai

panduan dalam proses penelitian pengenalan karakter huruf arab menggunakan metode kohonen.

3.3. Analisa & Perancangan

Tahapan ini merupakan proses analisis terhadap data-data yang telah berhasil dikumpulkan dalam penelitian pengenalan karakter huruf arab. Proses yang dilakukan dalam tahapan analisa dan perancangan metode kohonen *neural network* ini adalah dimulai dari tahapan pengenalan karakter seperti *data collection, pre-processing, processing* hingga menganalisanya dengan menggunakan metode kohonen *neural network.*

3.3.1. Analisa Kebutuhan Data

Penganalisaan data berupa data masukan dan keluaran sebagai pendukung implementasi dari metode. Penelitian ini membutuhkan data karakter huruf arab untuk proses pelatihan dari algoritma Kohonen *neural network.* Data karakter huruf arab didapat dari literatur mengenai jenis-jenis huruf arab. Data yang didapat akan dijadikan referensi pelatihan oleh aplikasi dan pengujian dilakukan dengan mengacu kepada hasil data (.dat) pelatihan yang telah disimpan.

3.3.2. Analisa Metode

Analisa metode akan menjelaskan mengenai beberapa tahapan yang dilakukan pada proses pengenalan karakter serta yang terdiri atas *pre-proecessing, processing, classification* (Kohonen *neural network*) dan juga penjelasan dari *euclidean distance.*

3.3.2.1.*Pre-Processing*

Tahapan ini dibagi menjadi 2 tahapan, yang pertama adalah proses konversi *image* dari format RGB *image* menjadi *image grayscale.* Tahapan selanjutnya adalah pengkoversian *image grayscale* menjadi *image* biner.

3.3.2.2.*Processing*

Processing adalah proses pencarian sebuah vektor dalam sebuah *image. Processing* memiliki beberapa tahapan dalam mengekstrak sebuah *image* vektor. Berikut adalah langkah-langkah dari tahapan *processing* :

3. Pemetaan ke area sampel
4. *Creating Vector*

3.3.2.3.Kohonen *Neural Network*

Dalam tahapan ini akan dijelaskan mengenai tahapan klasifikasi pengenalan karakter dari huruf arab menggunakan algoritma Kohonen *neural network* serta perhitungan menggunakan algoritma tersebut.

3.3.2.4.*Euclidean Distance*

Euclidean distance merupakan tahapan yang malakukan proses perbandingan vektor pada suatu pola. Pada tahapan ini akan ditentukan suatu pola apakah mirip atau tidak dengan pola yang diujikan.

3.3.3. Perancangan Aplikasi

Perancangan aplikasi terdiri atas *flowchart* dan perancangan antar muka aplikasi. Sub bab ini menjelaskan mengenai alur dari aplikasi yang akan dibangun melalui *flowchart* serta dari alur proses aplikasi yang ditunjukkan *flowchart* maka dilakukan perancangan antar muka aplikasi.

3.4. Impelementasi

Implementasi merupakan tahapan yang dilakukan setelah analisa dan perancangan selesai. Proses implementasi dilakukan menggunakan bahasa pemrograman java *standard edition. Tool* yang digunakan pada proses pemrograman adalah IDE Netbeans 7.1.Aplikasi yang dibuat dapat diimplementasikan pada *personal computer* ataupun *notebook* tanpa terkoneksi dengan jaringan internet.

3.5. Pengujian

Tahapan pengujian yang dilakukan pada aplikasi pengenalan karakter huruf arab bertujuan untuk mengetahui kesalahan-kesalahan yang terjadi pada proses pengenalan. Pengujian juga dilakukan untuk dapat mengetahui berapa persen-kah tingkat akurasi metode kohonen *neural network* untuk pengenalan karakter yang dilakukan oleh aplikasi terhadap karakter-karakter huruf arab yang dituliskanpada aplikasi tersebut. Pengujian dilakukan dengan metode *blackbox,* serta dilakukan juga pengujian *recognition* dan *error* yang dihitung dalam persentase tiap hurufnya.

3.6. Kesimpulan dan Saran

Tahapan kesimpulan dan saran adalah tahapan terakhir dari pelaksanaan tugas akhir. Tahapan ini berisi tentang kesimpulan dari hasil penelitian dan pengujian yang diperoleh serta saran-saran membangun yang dapat dijadikan panduan pada penelitian yang lebih baru agar dapat melakukan analisa serta implementasi metode kohonen *neural network* untuk pengenalan karakter huruf arab dengan lebih baik lagi.

BAB IV

ANALISA DAN PERANCANGAN

Analisa dan perancangan yang akan dilakukan adalah menganalisa mengenai tahapan pengenalan karakter serta menganalisa tahapan metode Kohonen *Neural Network* yang menguji tingkat akurasi dari metode tersebut dalam proses pengenalan karakter huruf Arab yang memiliki tingkat kerumitan yang lebih tinggi dibandingkan dengan karakter huruf lainnya. Bab ini akan fokus kepada tahapan pengenalan karakter, Kohonen *Neural Network,* parameter-parameter apa saja yang menjadi *input,* serta bagaimana proses *output* dihasilkan.

4.1. Analisa Kebutuhan Data

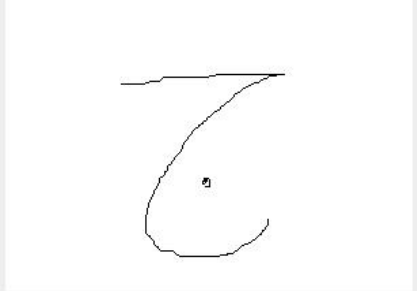
Untuk implementasi aplikasi pengenalan karakter dari metode Kohonen *Neural Network* dibutuhkan beberapa data masukan (*input*) dan data keluaran (*output*). Sub bab berikut akan membahas mengenai kebutuhan data tersebut.

4.1.1. Analisa Data Masukan (*Input*)

Beberapa data masukan (*input*) yang akan dianalisa dalam aplikasi pengenalan karakter ini adalah sebagai berikut :

1. Data pola karakter

   Data pola karakter adalah data masukan berupa data hasil penggambaran suatu pola karakter yang dilakukan pada suatu area kanvas gambar pada aplikasi pengenalan karakter. Hasil data pola karakter ini adalah berupa gambar dari pola-pola karakter huruf Arab.



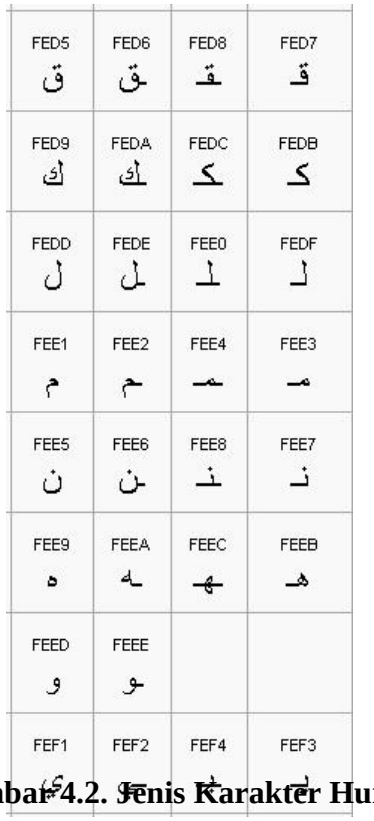
Gambar 4.1. Data Pola Karakter Yang Digambar Pada Area Kanvas

2. Data karakter huruf Arab

Data karakter adalah data berupa jenis huruf Arab yang dimasukkan oleh *user* setelah menggambarkan data bentuk pola karakter pada area kanvas penggambaran karakter. Data karakter-karakter huruf Arab ini dimasukkan dengan cara mengetikkan karakter hurufnya dengan menggunakan *keyboard.*

Isolated	End	Middle	Beginning
FE83 أ	FE84 ـأ		
FE8F ب	FE90 ـب	FE92 ـبـ	FE91 بـ
FE95 ت	FE96 ـت	FE98 ـتـ	FE97 تـ
FE99 ث	FE9A ـث	FE9C ـثـ	FE9B ثـ
FE9D ج	FE9E ـج	FEA0 ـجـ	FE9F جـ
FEA1 ح	FEA2 ـح	FEA4 ـحـ	FEA3 حـ
FEA5 خ	FEA6 ـخ	FEA8 ـخـ	FEA7 خـ
FEA9 د	FEAA ـد		
FEAB ذ	FEAC ـذ		
FEAD ر	FEAE ـر		

FEAF ز	FEB0 ـز		
FEB1 س	FEB2 ـس	FEB4 ـسـ	FEB3 سـ
FEB5 ش	FEB6 ـش	FEB8 ـشـ	FEB7 شـ
FEB9 ص	FEBA ـص	FEBC ـصـ	FEBB صـ
FEBD ض	FEBE ـض	FEC0 ـضـ	FEBF ضـ
FEC1 ط	FEC2 ـط	FEC4 ـطـ	FEC3 طـ
FEC5 ظ	FEC6 ـظ	FEC8 ـظـ	FEC7 ظـ
FEC9 ع	FECA ـع	FECC ـعـ	FECB عـ
FECD غ	FECE ـغ	FED0 ـغـ	FECF غـ
FED1 ف	FED2 ـف	FED4 ـفـ	FED3 فـ



Gambar 4.2. Jenis Karakter Huruf Arab

Dalam proses mendapatkan karakter huruf Arab, dilakukan proses konversi karakter huruf latin yang dimasukkan melalui *keyboard.* Tiap karakter huruf latin yang ada pada *keyboard* akan mempresentasikan satu karakter Arab, berikut adalah Tabel 4.1 konversi *keyboard* karakter huruf Arab ke latin :

Tabel 4.1. Konversi Huruf Arab *Isolated* (Terpisah)

Keyboard **karakter huruf latin**	**Hasil konversi huruf ke huruf Arab**
a	ا
b	ب
c	ت
d	ث
e	ج
f	ح
g	خ
h	د
i	ذ
j	ر
k	ز
l	س
m	ش
n	ص
o	ض
p	ط
q	ظ
r	ع
s	غ
t	ف
u	ق
v	ك
w	ل
x	م
y	ن
z	ه

1	و
2	ي

Tabel 4.2. Konversi Huruf Arab *Beginning* (Awal)

Keyboard **karakter huruf latin**	**Hasil konversi huruf ke huruf Arab**
3	بـ
4	تـ
5	ثـ
6	جـ
7	حـ
8	خـ
9	سـ
0	شـ
A	صـ
B	ضـ
C	طـ
D	ظـ
E	عـ
F	غـ
G	فـ
H	قـ
I	كـ
J	لـ
K	مـ

L	ـنـ
M	ـهـ
N	ـيـ

Tabel 4.3. Tabel Konversi Huruf Arab *Middle* (Tengah)

***Keyboard* karakter huruf latin**	**Hasil konversi huruf ke huruf Arab**
O	ـعـ
P	ـغـ
Q	ـفـ
R	ـقـ
S	ـهـ

Tabel 4.4. Konversi Huruf Arab *End* (Akhir)

***Keyboard* karakter huruf latin**	**Hasil konversi huruf ke huruf Arab**
T	ـأ
U	ـج
V	ـح
W	ـخ
X	ـع
Y	ـغ
Z	ـه

4.1.2. Analisa Proses

Ada beberapa tahapan dari proses pengenalan karakter yang diperoleh dari datamasukan yang telah dijelaskan sebelumnya. Berikut adalah tahapannya :

1. *User* mendaftarkan pola karakter huruf Arab dengan menggambarkan pola karakter pada area gambar (*canvas*) pada *form input character*, selanjutnya menentukan karakter dari pola huruf Arab yang baru saja digambar sesuai dengan daftar konversi huruf agar terdaftar pada *list* karakter yang telah dikenali.
2. Pola karakter-karakter yang telah ditambahkan kedalam *list* karakter yang dikenali lalu disimpan kedalam *file* .dat agar apabila aplikasi dijalankan lagi karakter-karakter yang telah dikenali dapat dimuat (*load*) kembali kedalam *list* karakter dari *file* .txt tersebut.
3. Pada *form recognize character*, karakter yang telah disimpan sebelumnya dimuat kedalam *list* karakter yang dikenali. Pada *list* karakter, karakter yang dikenali telah terkonversi menjadi karakter huruf Arab sesuai dengan daftar pada *form input character.*
4. Proses pengenalan karakter dilakukan dengan menggambarkan suatu pola karakter pada area gambar, hasil pengenalan akan ditampilkan melalui *dialogbox* yang menyampaikan pola karakter yang digambarkan menyerupai suatu karakter hururf Arab tertentu.

4.1.3. Analisa Data Keluaran (*Output*)

Tujuan dari penelitian ini adalah menguji persentase dari metode Kohonen *Neural Network* untuk pengenalan karakter dengan studi kasus karakter huruf Arab. Aplikasi akan menampilkan suatu karakter yang akan diuji apakah sama dengan pola yang telah didapat sebelumnya. Data keluaran akan ditampilkan dalam persentase kemiripan dan tingkat akurasi data pengujian dengan pola yang telah didapat dari proses *training.*

4.2. Analisa Metode

Analisa metode akan menjelaskan mengenai beberapa tahapan yang dilakukan pada proses pengenalan karakter serta yang terdiri atas *pre-proecessing, processing, classification* (Kohonen *Neural Network*) hingga penjelasan dari perhitungan *euclidean distance.*

4.2.1. Analisa *Pre-Processing*

Proses *pre-processing* digunakan untuk mengubah format pola karakter yang digambar yang memiliki format RGB menjadi *grayscale.* Berikut adalah langkah-langkah mengubah pola karakter yang memiliki formatRGB menjadi *grayscale* menggunakan algoritma Otsu.

4.2.1.1.Konversi Citra ke *Grayscale*

Citra yang akan diproses akan diubah terlebih dahulu kedalam format *grayscale* dengan cara diambil nilai R, G dan B dari masing-masing *pixel* citra dengan menggunakan rumus :

Grayscale = R 1/30 + G 1/59 + B 1/11

Berikut adalah *pseudocode* untuk konversi pola karakter huruf Arab dari format RGB menjadi menjadi *grayscale* :

```
Inisialisasi Height
Inisialisasi Width
Inisialisasi GrayScaleImg [Height] [Width]
Do i = 0 to Height – 1
        Do j = 0 to Width – 1
                GrayScaleImg [i] [j] = Image [i] [j] .R * 0,30 + Image [i] [j]  .G * 0,59 + Image [i] [j]
                .B * 0,11
        EndDo
EndDo
```

Algoritma 4.1. Proses Konversi ke *Grayscale*

4.2.1.2.Konversi *Grayscale* ke *Bynary Image*

Setelah proses konversi ke format *grayscale* proses selanjutnya adalah mengubahnya menjadi format biner. Namun, sebelumnya dicari terlebih dahulu nilai *threshold* dari histogram nilai *grayscale.* Dengan cara mencari dari histogram di nilai *grayscale* mana yang terjadi perubahan instensitas yang cukup signifikan (kontras), maka diambil nilai tengah dari nilai *grayscale* tersebut sebagai nilai *threshold.*

Berikut adalah *pseudocode* untuk mencari nilai *threshold* dari format *grayscale* suatu pola karakter :

```
Inisialisasi Histogram
Inisialisasi ContrastHistogram
Inisialisasi Height
Inisialisasi Width
Inisialisasi variabel-variabel
Do i = 0 to Height – 1
        Do j = 0 Width – 1
                Pengisian Histogram
                Cek maksimum konstras
        EndDo
```

```
EndDo
Do i = 0 to Height – 1
        Do j = 0 to Width – 1
                Pengisian ConstrastHistogram dengan nilai konstras yang relevan
        EndDo
EndDo
Inisialisasi Sum, ContrastSum, ContrastMass, IMass
Do i = 0 to 255
        Sum = Sum + i * ContrastHistogram [i]
        ContrastMass = ContrastMass + ContrastHistogram [i]
        IMass = IMass + Histogram [i]
EndDo
```

```
Inisialisasi MassKiri, MassKanan, MidKiri, MidKanan, Fmax, OtsuValue
Do i = 0 to 255
        MassKiri = MassKiri + ContrastHistogram [i]
        If MassKiri = 0 then Continue
        MassKanan = ContrastMass – MassKiri
        If MassKanan = 0 then Break
        ContrastMass = ContrastMass + i * ContrastHistogram [i]
        MidKiri = ContrastMass / MassKiri
        MidKanan = (Sum – ContrastSum) / MassKanan
        OtsuValue = MassKiri * MassKanan * (MidKiri - MidKanan)
        If OtsuValue > Fmax then
                Fmax = OtsuValue
                Threshold = i + 1
        EndIf
EndDo
```

Algoritma 4.2. Pencarian Nilai *Threshold*

Setelah mendapatkan nilai *threshold* maka selanjutnya adalah mengubah nilai tersebut ke format biner dengan kondisi jika nilai *grayscale* dari *pixel* dibawah nilai *threshold*, maka *pixel* menjadi *pixel* hitam, dan jika nilai *grayscale* dari *pixel* diatas atau sama dengan nilai *threshold*, maka *pixel* menjadi *pixel* putih. Berikut adalah *pseudocode*-nya :

```
Inisialisasi BWImage
Inisialisasi Height
Inisialisasi Width
Inisialisasi Threshold = OtsuThreshold()
Do i = 0 to Height – 1
        Do j = 0 to Width – 1
                If Image [i] [j] >= Threshold then
                        BWImage [i] [j] = 255
                ElseIf Image [i] [j] < Threshold then
                        BWImage [i] [j] = 0
                EndIf
```

EndDo

EndDo

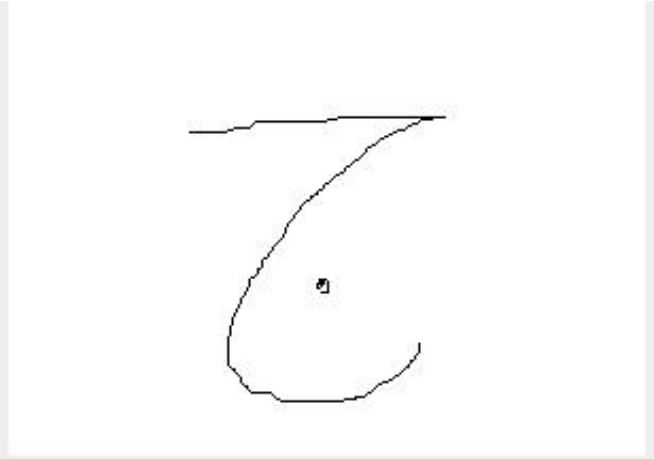
Algoritma 4.3. Konversi ke Format Biner

4.2.2. Analisa *Processing*

Processing merupakan tahapan yang terdiri dari proses pemetaan bilangan biner kedalam area sampel dan menciptakan vektor dari sampel tersebut. Berikut penjelasan dari analisa kedua proses tersebut.

4.2.2.1.Pemetaan ke Area Sampel

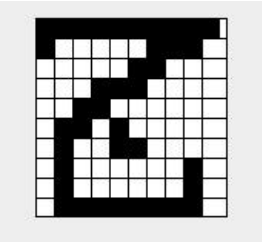
Setelah mendapatkan nilai biner dari pola yang digambar, maka hal yang selanjutnya dilakukan adalah memetakan pola karakter pada area kanvas gambar dan melakukan proses pengubahan ukuran (*rescale image*) yang semula berukuran 100x100 *pixel* menjadi matriks berukuran 10x10. Misalkan yang digambar adalah pola karakter huruf ج (jim) sebagai berikut :



Gambar 4.3. Pola Karakter Yang Digambar Pada Kanvas Berukuran 100 x 100 *Pixel*

Apabila pola karakter telah digambar, maka pola karakter huruf ج (jim) dipetakan kedalam matriks berukuran 10 x 10 seperti yang ditunjukkan pada gambar

4.4. Pada matriks dibawah ini titik matriks yang berwarna hitam memiliki nilai biner 1 dan titik matriks yang berwarna putih memiliki nilai biner 0.



Gambar 4.4. Data Pola Karakter Dengan Ukuran Matriks 10 x 10 *pixel*

4.2.2.2.*Creating Vector*

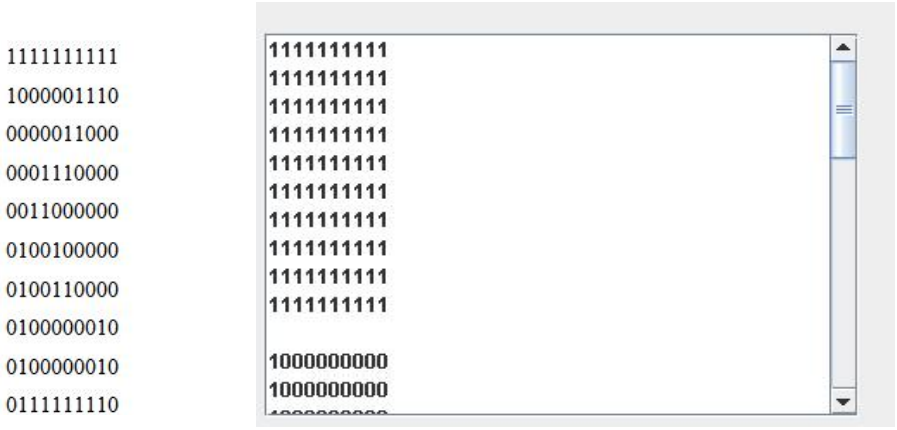
Nilai biner dari pola karakter akan disimpan pada *file* yang berformat .dat dan berfungsi untuk pencocokan pola karakter yang akan duiji nantinya.

Tabel 4.5. Konversi Biner Pola Karakter

1 **[1,1]**	**1** **[1,2]**	**1** **[1,3]**	**1** **[1,4]**	**1** **[1,5]**	**1** **[1,6]**	**1** **[1,7]**	**1** **[1,8]**	**1** **[1,9]**	**1** **[1,10]**
1 **[2,1]**	0 [2,2]	0 [2,3]	0 [2,4]	0 [2,5]	0 [2,6]	**1** **[2,7]**	**1** **[2,8]**	**1** **[2,9]**	0 [2,10]
0 [3,1]	0 [3,2]	0 [3,3]	0 [3,4]	0 [3,5]	**1** **[3,6]**	**1** **[3,7]**	0 [3,8]	0 [3,9]	0 [3,10]
0 [4,1]	0 [4,2]	0 [4,3]	**1** **[4,4]**	**1** **[4,5]**	**1** **[4,6]**	0 [4,7]	0 [4,8]	0 [4,9]	0 [4,10]
0 [5,1]	0 [5,2]	**1** **[5,3]**	**1** **[5,4]**	0 [5,5]	0 [5,6]	0 [5,7]	0 [5,8]	0 [5,9]	0 [5,10]
0 [6,1]	**1** **[6,2]**	0 [6,3]	0 [6,4]	**1** **[6,5]**	0 [6,6]	0 [6,7]	0 [6,8]	0 [6,9]	0 [6,10]
0 [7,1]	**1** **[7,2]**	0 [7,3]	0 [7,4]	**1** **[7,5]**	**1** **[7,6]**	0 [7,7]	0 [7,8]	0 [7,9]	0 [7,10]
0	**1**	0	0	0	0	0	0	**1**	0

[8,1]	**[8,2]**	[8,3]	[8,4]	[8,5]	[8,6]	[8,7]	[8,8]	**[8,9]**	[8,10]
0 [9,1]	**1 [9,2]**	0 [9,3]	0 [9,4]	0 [9,5]	0 [9,6]	0 [9,7]	0 [9,8]	**1 [9,9]**	0 [9,10]
0 [10,1]	**1 [10,2]**	**1 [10,3]**	**1 [10,4]**	**1 [10,5]**	**1 [10,6]**	**1 [10,7]**	**1 [10,8]**	**1 [10,9]**	0 [10,10]

Dari tabel di atas dapat dilihat nilai biner 1 yang ditebalkan merupakan hasil konversi biner dari pola karakter ج (jim) dengan bobot matriks [i,j] 10 x 10. Pada aplikasi, hasil dari konversi biner ini dipresentasikan pada form *recognize character* seperti berikut.



Gambar 4.5. Representasi Biner Pola Karakter ج (jim)

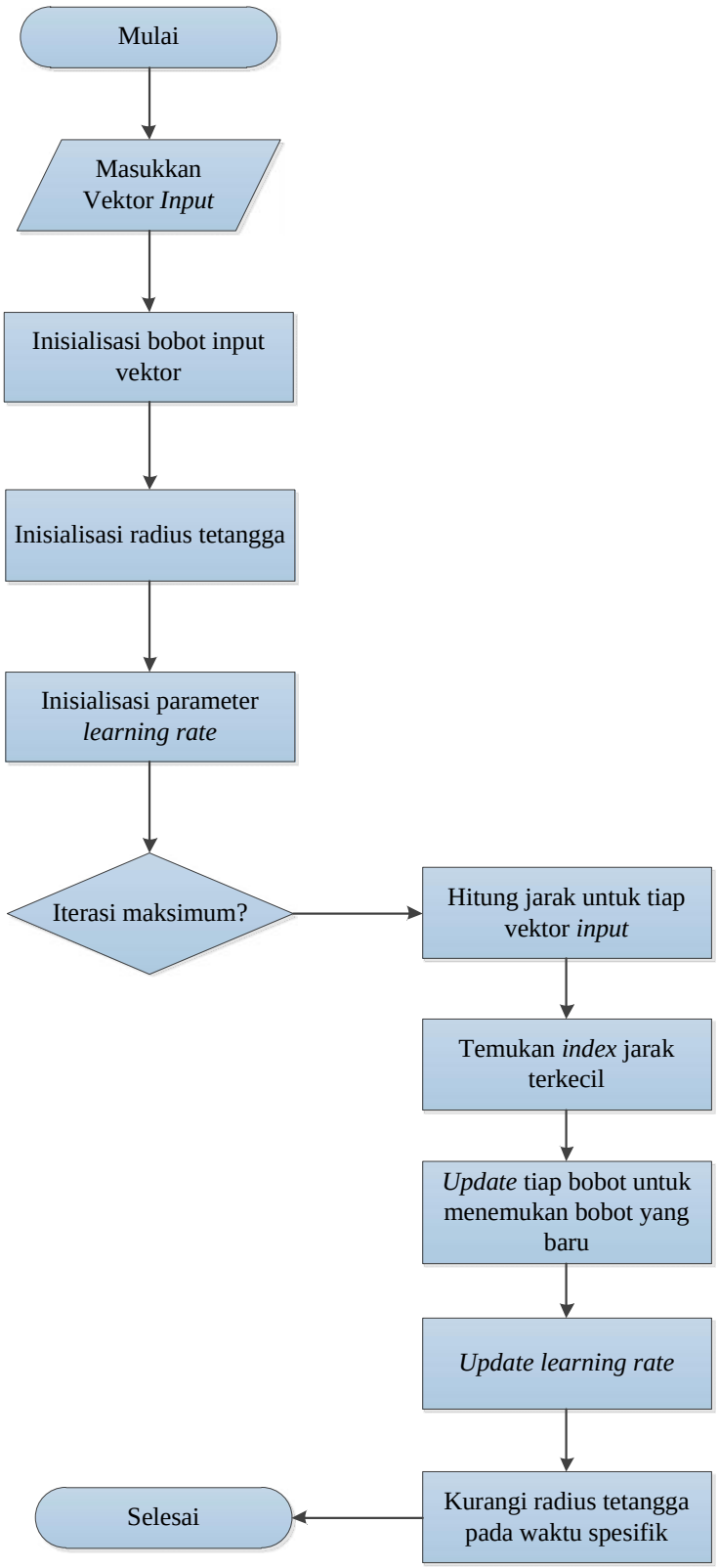
Representasi bilangan biner dari pola karakter juga dapat dilihat didalam *file sample.*dat yang disimpan sebelumnya pada folder aplikasi.

e:1111111111100000111000000110000001110000001100000000100100000010011000001000000100100000010011111111 0

Gambar 4.6. Representasi Biner pada Sample.dat

4.2.3. Analisa Kohonen *Neural Network*

Metode Kohonen *Neural Network* melakukan prosesnya dengan cara membandingkan data bobot matriks vektor karakter pembanding dengan bobot matriks vektor yang didapat dari hasil penggambaran yang telah diubah ukurannya kedalam ukuran matriks 10*10 yang kemudian disimpan pada *sample.*dat. Berikut adalah *flowchart* dari proses Kohonen *Neural Network* seperti tahapan yang telah dijelaskan pada bab 2.


Mulai
Masukkan Vektor *Input*
Inisialisasi bobot input vektor
Inisialisasi radius tetangga
Inisialisasi parameter *learning rate*
Iterasi maksimum?
Hitung jarak untuk tiap vektor *input*
Temukan *index* jarak terkecil
Update tiap bobot untuk menemukan bobot yang baru
Update learning rate
Kurangi radius tetangga pada waktu spesifik
Selesai

Gambar 4.7. *Flowchart* Algoritma Kohonen *Neural Network*

4.2.3.1. Contoh Perhitungan Menggunakan Kohonen *Neural Network*

Berikut adalah sebuah contoh inisialisasi sampel vektor yang dibagi kedalam 4 vektor *input* : X_1 = [1,1,0,0], X_2 = [0,0,0,1], X_3 = [1,0,0,0], X_4 = [0,0,1,1]. Jumlah maksimum dari *cluster* yang disajikan adalah : m = 2. Dan memiliki *learning rate* α = 0,6.

Dengan hanya ada 2 *cluster* yang tersedia, maka radius tetangga dari *node* J diatur sehingga hanya satu *cluster* yang melakukan proses *update* bobot pada tiap langkah yang berarti juga memiliki radius ketetanggaan R = 0.

a. Langkah 0, inisialisasi bobot matriks w_{ij} :

$$\begin{pmatrix} 0,2 & 0,8 \\ 0,6 & 0,4 \\ 0,5 & 0,7 \\ 0,9 & 0,3 \end{pmatrix}$$

Inisialisasi radius tetangga :

R = 0

Inisialisasi *learning rate* :

α(0) = 0,6

b. Langkah 1, lakukan proses *training*

c. Langkah 2, untuk vektor pertama (1,1,0,0), lakukan langkah 3-5

d. Langkah 3,

$$D(1) = (0,2 - 1)^2 + (0,6 - 1)^2 + (0,5 - 0)^2 + (0,9 - 0)^2$$

$$= 0,64 + 0,16 + 0,25 + 0,81$$

$$= 1,86$$

$$D(2) = (0,8 - 1)^2 + (0,4 - 1)^2 + (0,7 - 0)^2 + (0,3 - 0)^2$$

$= 0,04 + 0,36 + 0,49 + 0,09$

$= 0,98$

e. Langkah 4, vektor *input* yang memiliki nilai lebih kecil adalah *output node* 2, sehingga J = 2

f. Langkah 5, lakukan *update* bobot pada *unit* pemenang

$w_{i2}(new) = w_{i2}(old) + 0,6\ [x_i - w_{i2}(old)]$

$= [0,8\ 0,4\ 0,7\ 0,3] + 0,6\ [(1\ 1\ 0\ 0) - (0,8\ 0,4\ 0,7\ 0,3)]$

$= [0,92\ \ 0,76\ \ 0,28\ \ 0,12]$

Perhitungan di atas akan menghasilkan bobot matriks sebagai berikut :

$$\begin{pmatrix} 0,2 & 0,92 \\ 0,6 & 0,76 \\ 0,5 & 0,28 \\ 0,9 & 0,12 \end{pmatrix}$$

g. Langkah 2, untuk vektor kedua (0,0,0,1) lakukan langkah 3-5

h. Langkah 3,

$D(1) = (0,2 - 0)^2 + (0,6 - 0)^2 + (0,5 - 0)^2 + (0,9 - 1)^2$

$= 0,04 + 0,36 + 0,25 + 0,01$

$= 0,66$

$D(2) = (0,92 - 0)^2 + (0,76 - 0)^2 + (0,28 - 0)^2 + (0,12 - 1)^2$

$= 0,8464 + 0,5776 + 0,0784 + 0,7744$

$= 2,2768$

i. Langkah 4, vektor *input* yang memiliki nilai lebih kecil adalah *output node* 1, sehingga J = 1

j. Langkah 5, lakukan *update* bobot pada *unit* pemenang

$w_{i1}(new) = w_{i1}(old) + 0,6\ [x_i - w_{i1}(old)]$

= [0,2 0,6 0,5 0,9] + 0,6 [(0 0 0 1)
– (0,2 0,6 0,5 0,9)]
= [0,08 0,24 0,2 0,96]

Perhitungan di atas akan menghasilkan bobot matriks sebagai berikut :

$$\begin{pmatrix} 0,08 & 0,92 \\ 0,24 & 0,76 \\ 0,2 & 0,28 \\ 0,96 & 0,12 \end{pmatrix}$$

k. Langkah 2, untuk vektor ketiga (1,0,0,0) lakukan langkah 3-5

l. Langkah 3,

$D(1) = (0,08 - 1)^2 + (0,24 - 0)^2 + (0,2 - 0)^2 + (0,96 - 0)^2$

$= 1,8656$

$D(2) = (0,92 - 1)^2 + (0,76 - 0)^2 + (0,28 - 0)^2 + (0,12 - 0)^2$

$= 0,6768$

m. Langkah 4, vektor *input* yang memiliki nilai lebih kecil adalah *output node* 2, sehingga J = 2

n. Langkah 5, lakukan *update* bobot pada *unit* pemenang

$w_{i2}(new) = w_{i2}(old) + 0,6\,[x_i - w_{i2}(old)]$

= [0,92 0,76 0,28 0,12] + 0,6 [(1 0 0 0) – (0,92 0,76 0,28 0,12)]

= [0,968 0,304 0,112 0,048]

Perhitungan di atas akan menghasilkan bobot matriks sebagai berikut :

$$\begin{pmatrix} 0,08 & 0,968 \\ 0,24 & 0,304 \end{pmatrix}$$

0,2 0,112

0,96 0,048

o. **Langkah 2,** untuk vektor keempat (0,0,1,1) lakukan langkah 3-5

p. **Langkah 3,**

$$D(1) = (0,08 - 0)^2 + (0,24 - 0)^2 + (0,2 - 1)^2 + (0,96 - 1)^2$$

$$= 0,7056$$

$$D(2) = (0,968 - 0)^2 + (0,304 - 0)^2 + (0,112 - 1)^2 + (0,048 - 1)^2$$

$$= 2,724$$

q. **Langkah 4,** vektor *input* yang memiliki nilai lebih kecil adalah *output node* 1, sehingga J = 1

r. **Langkah 5,** lakukan *update* bobot pada *unit* pemenang

$$w_{i1}(new) = w_{i1}(old) + 0,6\,[x_i - w_{i1}(old)]$$

= [0,08 0,24 0,2 0,96] + 0,6 [(0 0 1 1) – (0,08 0,24 0,2 0,96)]

= [0,032 0,096 0,68 0,984]

Perhitungan di atas akan menghasilkan bobot matriks sebagai berikut :

$$\begin{pmatrix} 0,032 & 0,968 \\ 0,096 & 0,304 \\ 0,68 & 0,112 \\ 0,984 & 0,048 \end{pmatrix}$$

s. **Langkah 6,** *update learning rate* :

A = 0,5 (0,6) = 0,3

4.2.4. Analisa *Euclidean Distance*

Untuk mengetahui jarak terbaik dari proses pengenalan pola karakter, maka dilakukan perhitungan dengan menggunakan *Euclidean Distance* dengan membandingkan matriks bobot dari perhitungan terakhir metode Kohonen. *Euclidean distance* dipilih dikarenakan menurut pelitian perbandingan antara beberapa metode jarak yang dilakukan oleh Syamani (2008), metode *city block / Manhattan distance* dan beberapa lainnya lebih unggul dalam proses kecepatan. Namun dalam hal akurasi *euclidean distance* sedikit lebih unggul dibanding yang lainnya, selain itu *euclidean distance* juga lebih banyak digunakan dalam berbagai penelitian.

Berikut adalah contoh matriks yang diambil dari perhitungan langkah ke-5 pada sub-bab sebelumnya :

A = [0,032, 0,096, 0,68, 0,984]

B = [0,969, 0,304, 0,112, 0,048]

merujuk pada rumus 2.6, *euclidean distance* dari vektor A dan B adalah :

$d_{ij} = \sqrt{((0{,}032 - 0{,}969)^2 + (0{,}096 - 0{,}304)^2 + (0{,}68 - 0{,}112)^2 + (0{,}984 - 0{,}048))^2}$

$= \sqrt{(-0{,}937)^2 + (-0{,}208)^2 + (0{,}568)^2 + (0{,}936)^2}$

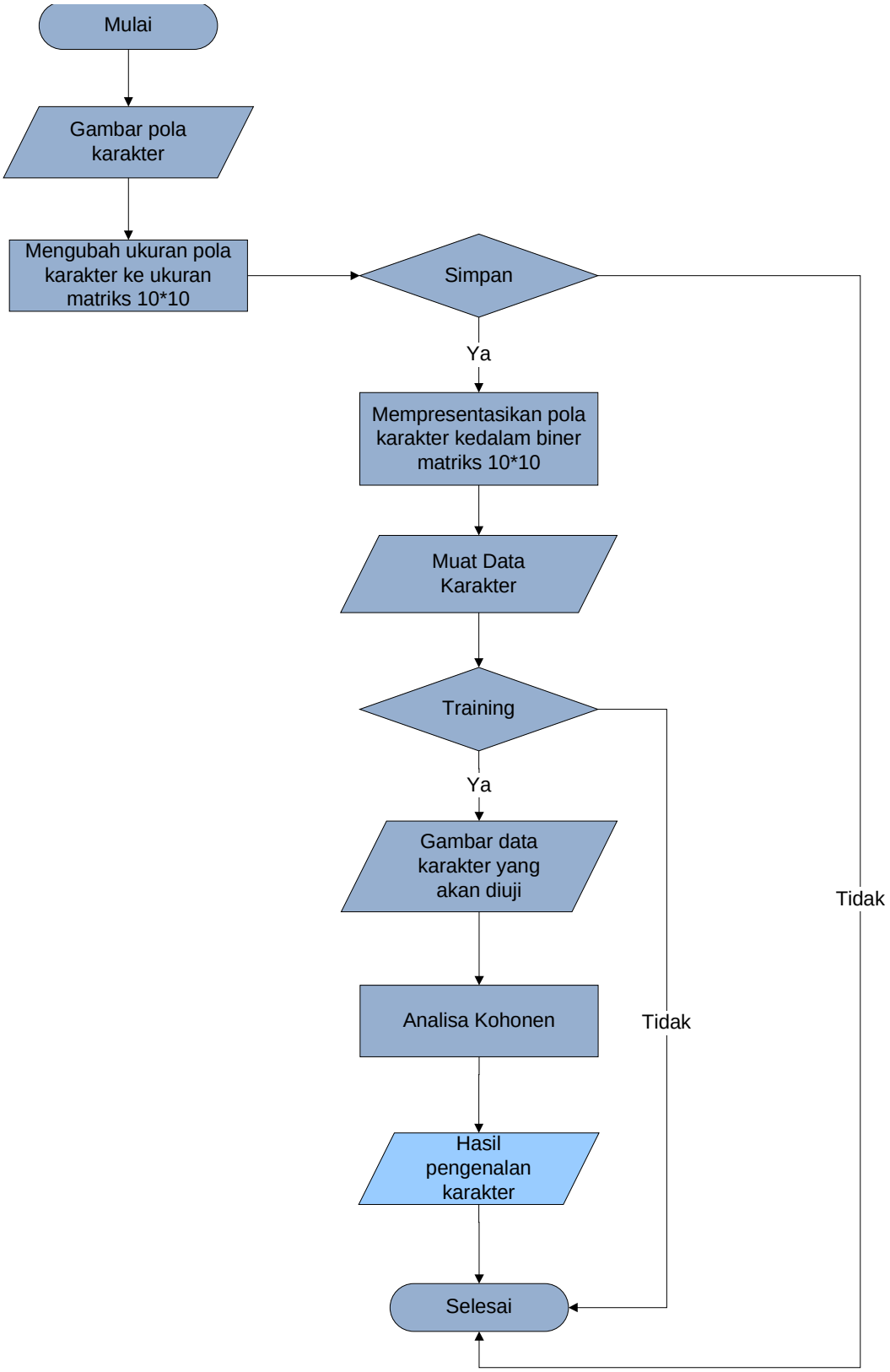
$= \sqrt{(0{,}877) + (0{,}043) + (0.003) + (0{,}876)}$

= 1.341

4.3. Perancangan Aplikasi

4.3.1. *Flowchart*

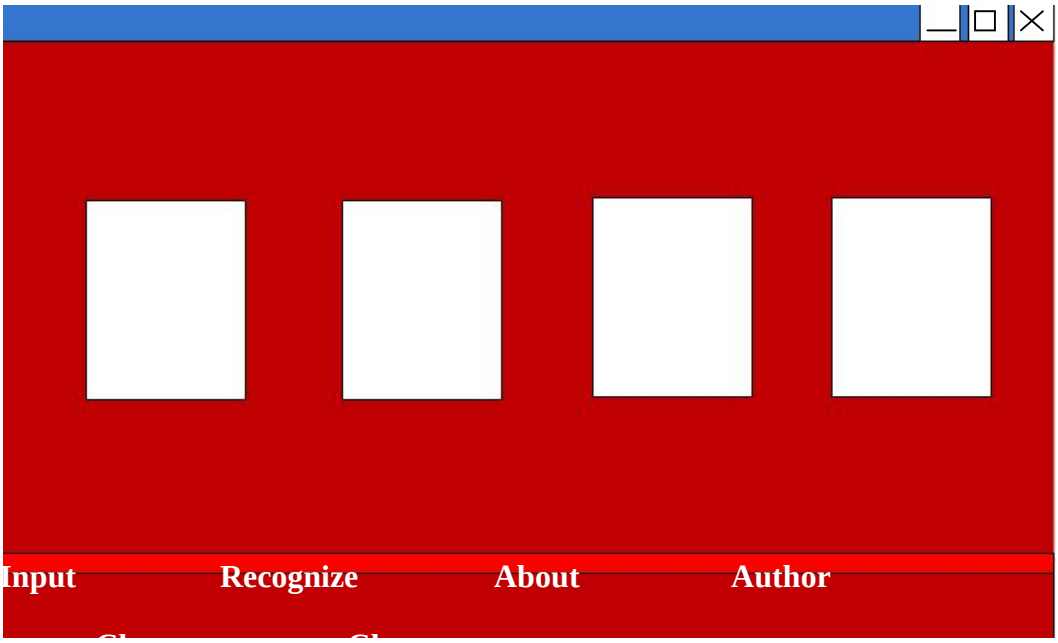
Tahapan dari proses aplikasi pengenalan karakter menggunakan metode Kohonen *Neural Network* ini dapat digambarkan melalui *flowchart* sebagai berikut:


Mulai
Gambar pola karakter
Mengubah ukuran pola karakter ke ukuran matriks 10*10
Simpan
Ya
Tidak
Mempresentasikan pola karakter kedalam biner matriks 10*10
Muat Data Karakter
Training
Ya
Tidak
Gambar data karakter yang akan diuji
Analisa Kohonen
Hasil pengenalan karakter
Selesai

Gambar 4.8. *Flowchart* Aplikasi Pengenalan Karakter Huruf Arab

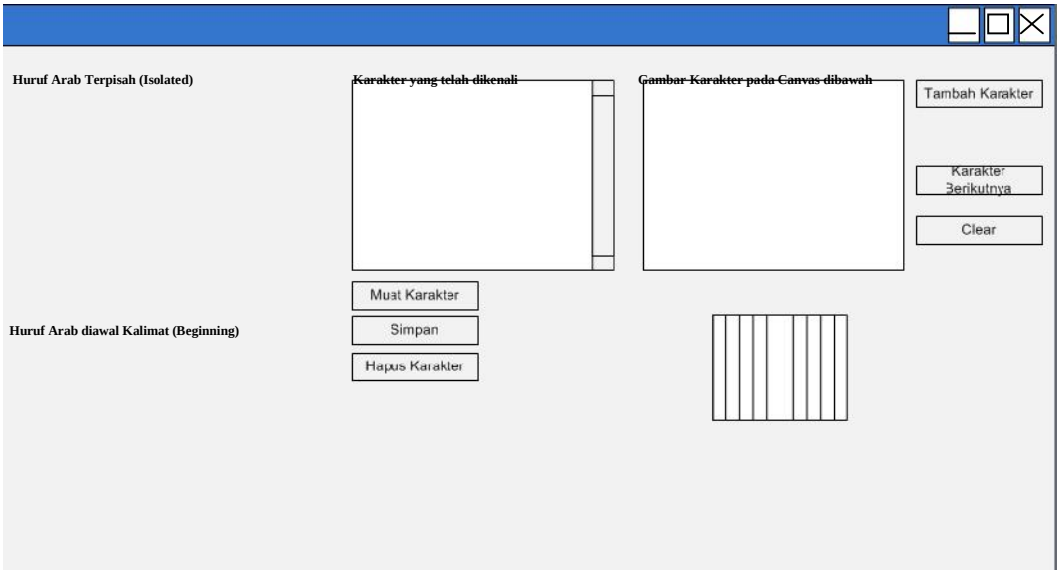
4.3.2. Perancangan Antar Muka

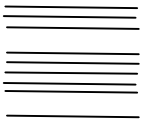
4.3.2.1.Perancangan Antar Muka *Form* Utama



Gambar 4.9. Antar Muka *Form* Utama

4.3.2.2.Perancangan Antar Muka *Form Input Character*



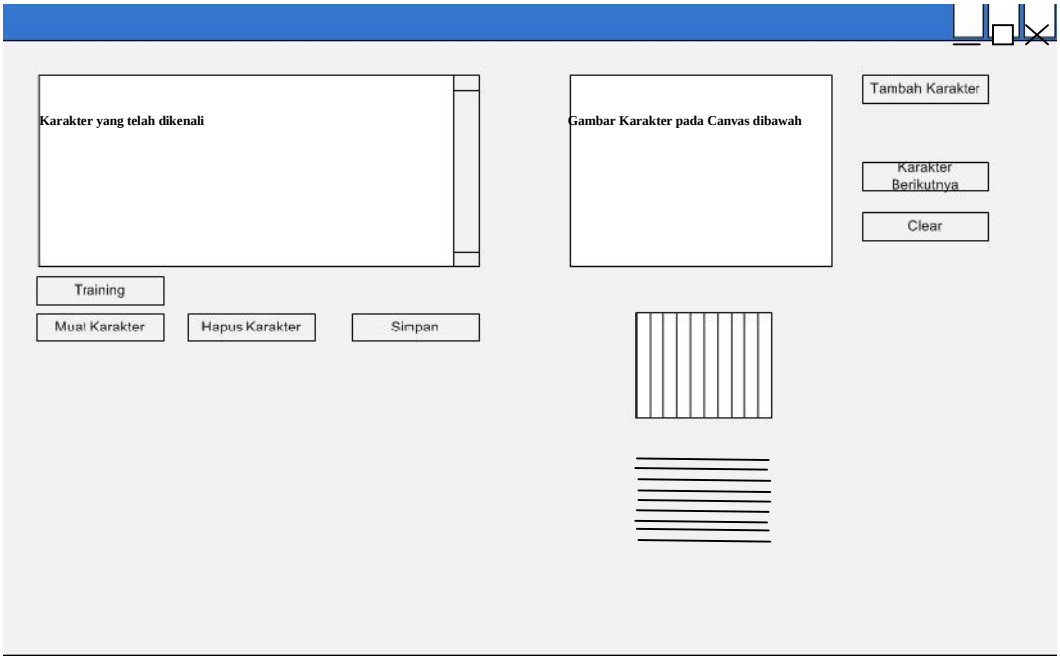


Huruf Arab ditengah Kalimat (Middle)

Huruf Arab diakhir Kalimat (End)

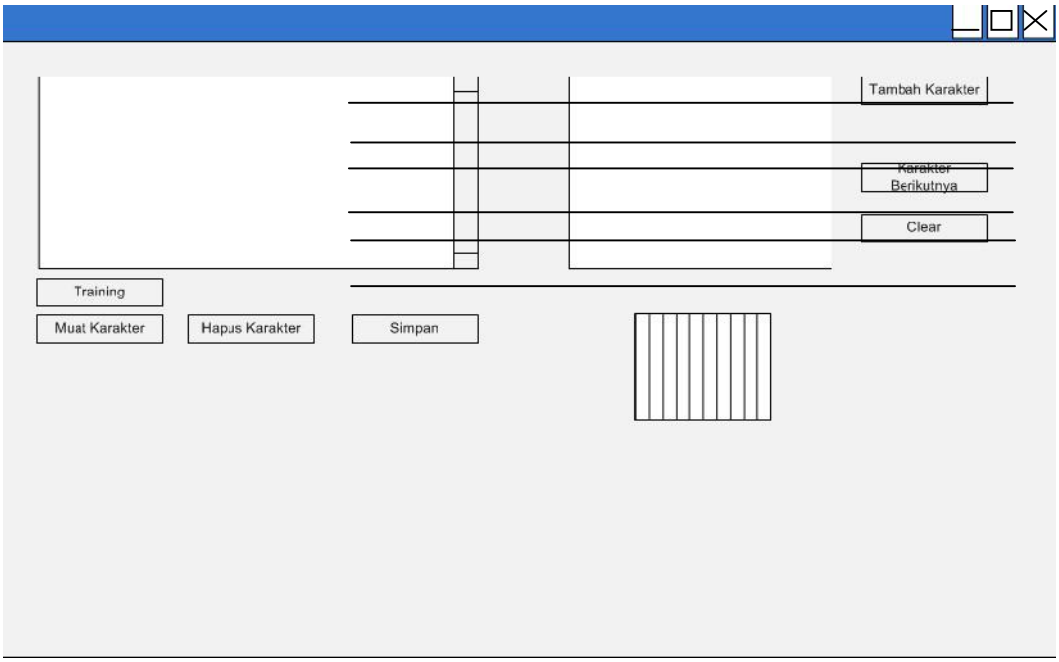
Gambar 4.10. Antar Muka *Form Input Character*

4.3.2.3.Perancangan Antar Muka *FormRecognize Character*



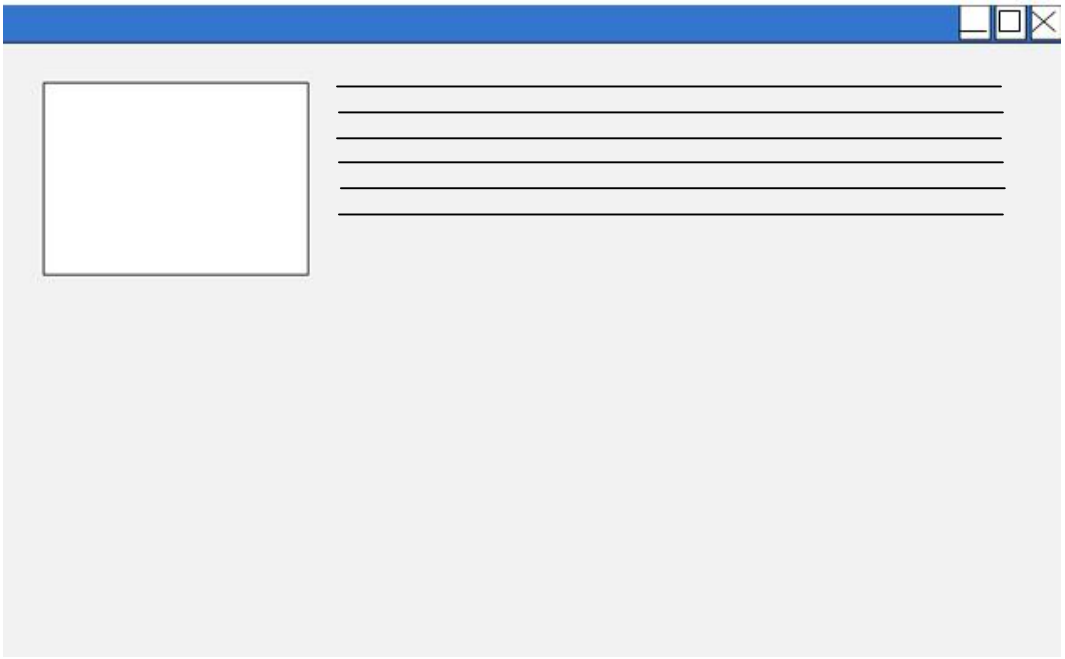
Gambar 4.11. Antar Muka *FormRecognize Character*

4.3.2.4.Perancangan Antar Muka *Form About*



Gambar 4.12. Antar Muka *Form About*

4.3.2.5.Perancangan Antar Muka *FormAuthor*



Gambar 4.13. Antar Muka *FormAuthor*

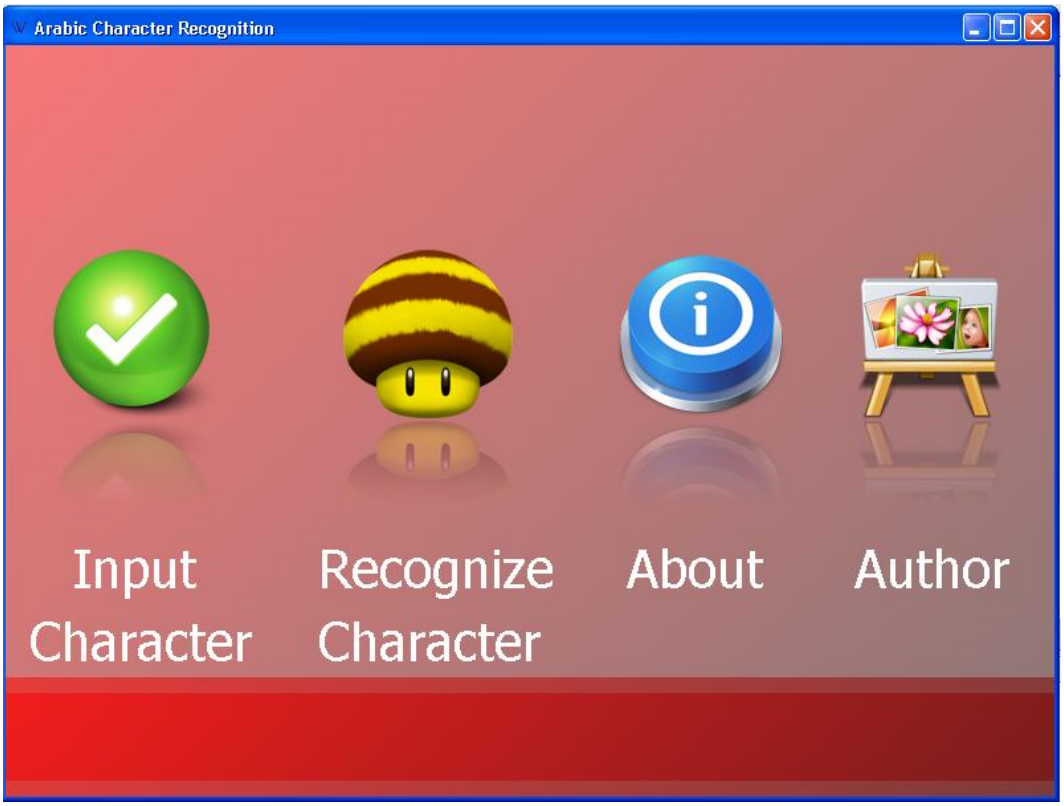
BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1. Implementasi

Aplikasi *Arabic character recognition* menggunakan metode Kohonen ini diimplementasikan dengan menggunakan bahasa pemrograman Java *Standard Edition* yang berjalan pada *platform desktop* untuk memudahkan proses pengujian aplikasi, sehingga dapat dijalankan tanpa koneksi internet.

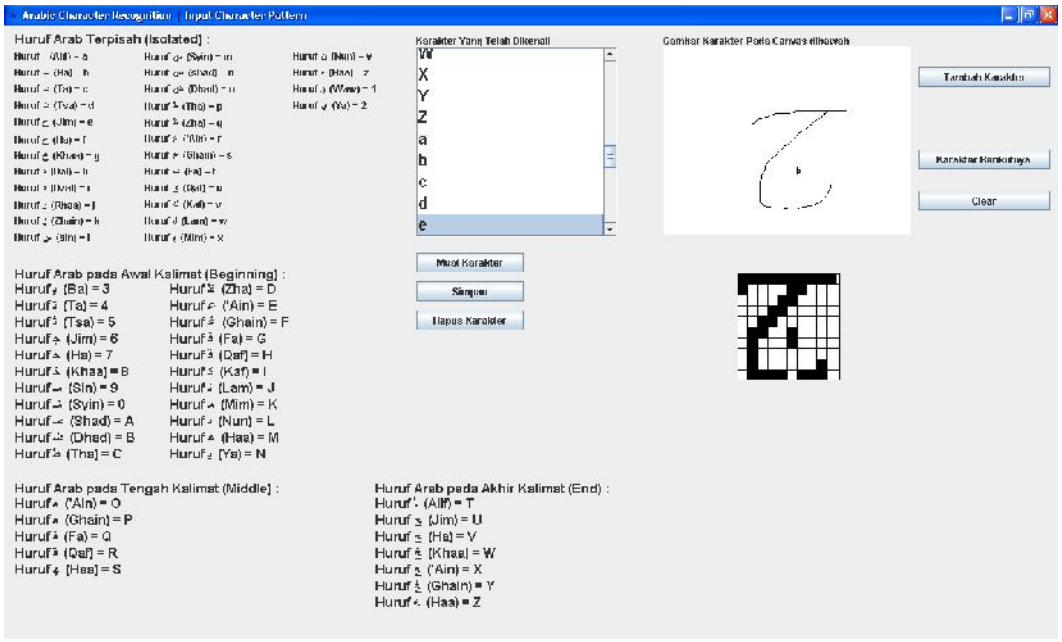
5.1.1. Implementasi *Form* Utama



Gambar 5.1. *Form* Utama

Terdapat 4 menu utama pada *form* utama, *form input character*, *form recognize character*, *form about* dan *form author*. Rincian menu akan dijelaskan pada sub bab berikutnya.

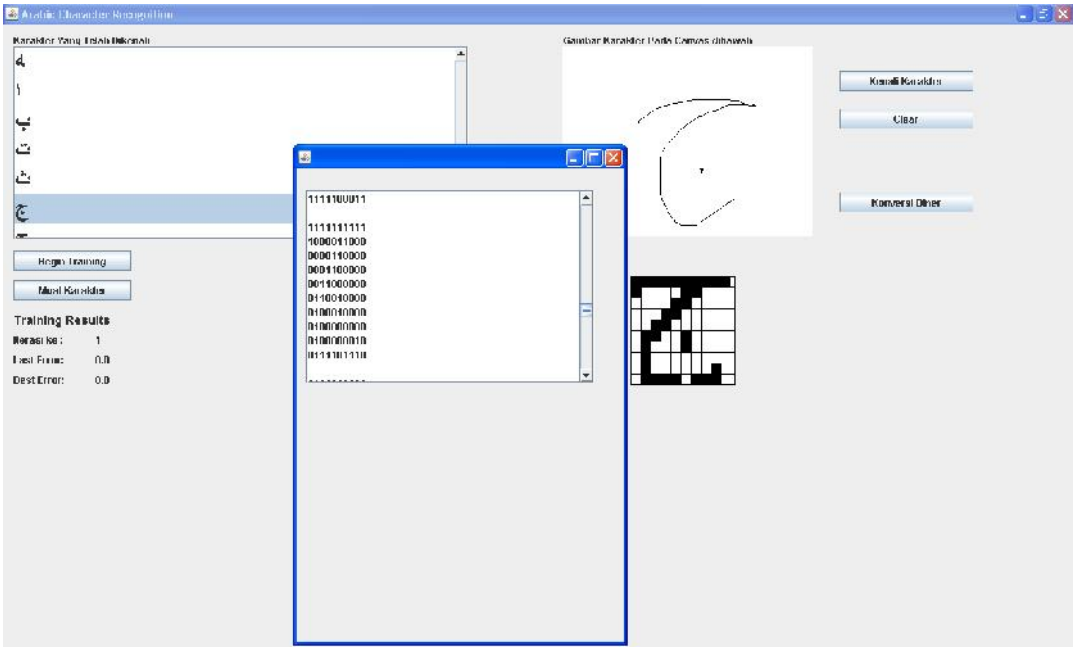
5.1.2. Implementasi *Form Input Character*



Gambar 5.2. *Form Input Character*

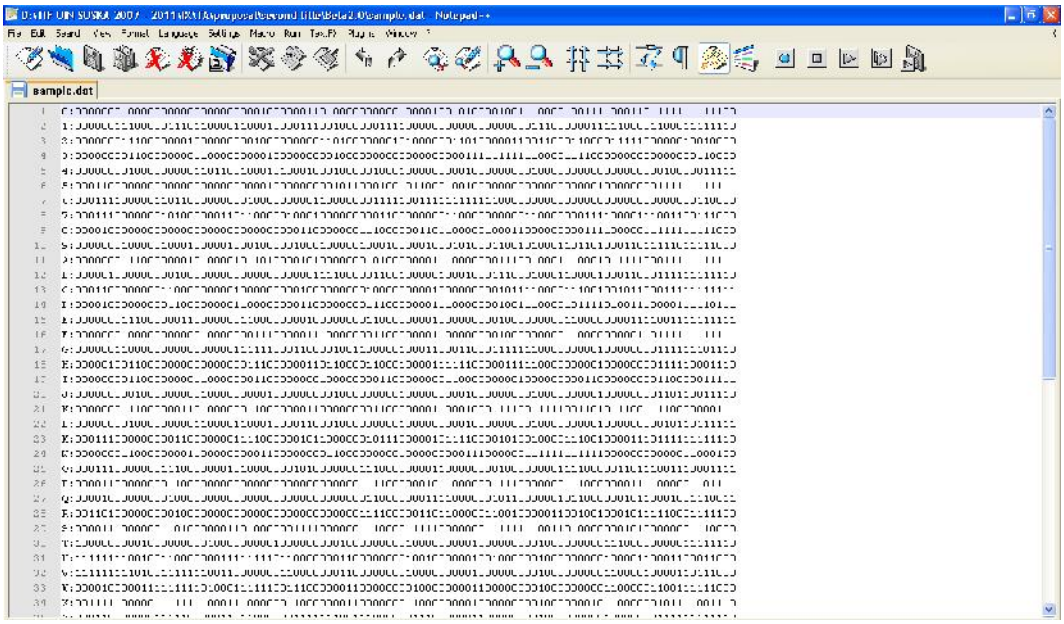
Pada sisi kiri hingga dibawah *form input character* terdapat daftar yang menunjukkan konversi *input* data huruf arab dengan menggunakan karakter yang ada pada *keyboard* huruf latin. Apabila *user* ingin memasukkan suatu karakter huruf arab maka terlebih dahulu melihat pada daftar huruf apa yang harus diketik pada *keyboard* untuk proses konversinya menjadi huruf arab. Matriks 10*10 akan ada apabila *user* ingin menambahkan pola karakter yang digambar kedalam karakter yang dikenali dengan melakukan klik pada tombol tambah karakter. Karakter-karakter yang telah ada disimpan sebelumnya bisa kembali dimuat dengan malakukan klik pada tombol muat karakter. Pada *form* ini terdapat proses *preprocessing*,*processing* serta penyimpanan pola karakter yang akan diuji pada *file sample*.dat.

5.1.3. Implementasi *FormRecognize Character*



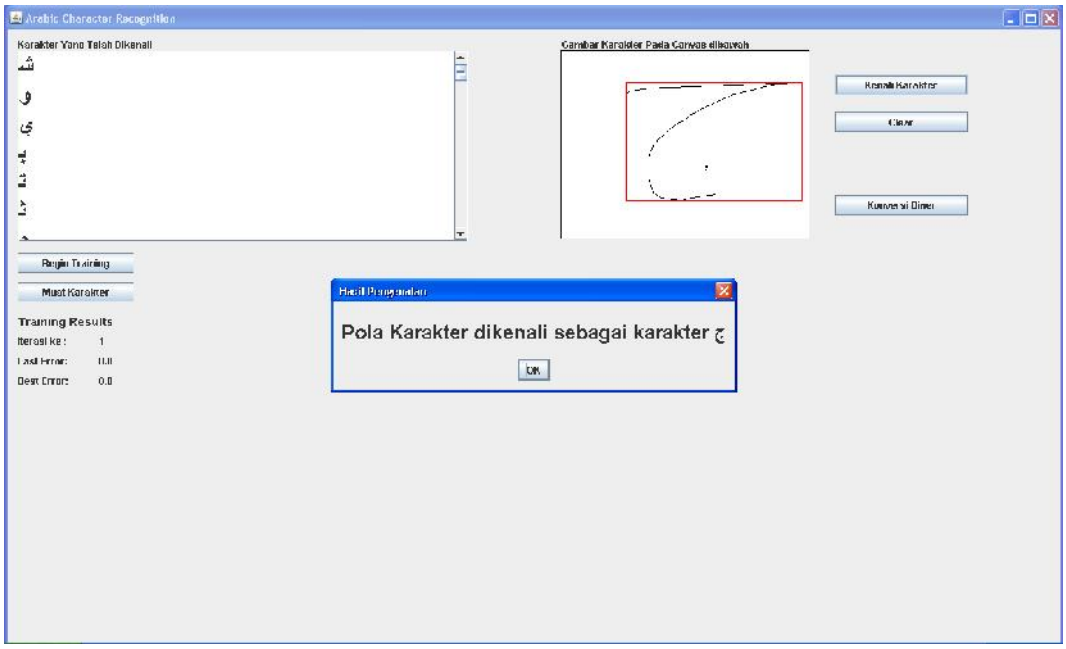
Gambar 5.3. *Form Recognize Character*

Form recognize character akan menampilkan *list* karakter huruf arab yang sebelumnya telah disimpan pada *file sample*.dat dengan menekan tombol muat karakter. Proses *training* karakter akan dimulai dengan menggambarkan pola karakter yang akan duiji pada area kanvas pada kanan atas dilanjutkan dengan menekan tombol *training.* Pengujian karakter dilakukan dengan menekan tombol kenali karakter. Representasi biner dari pola karakter yang telah disimpan dapat dilihat dengan melakukan klik pada tombol Konversi biner ataupun dengan melihat langsung pada *file sample*.dat seperti dibawah ini.



Gambar 5.4. Representasi Biner *Sample*.dat

Bila melihat dari *file sample.dat* akan terlihat karakter huruf-huruf latin dan angka pada huruf pertama dari baris pertama hingga terakhir. Huruf dan angka latin ini pada aplikasi dikonversi menjadi karakter arab pada *form recognize character.*



Gambar 5.5. Hasil Pengenalan Karakter Huruf ج (Jim)

Untuk menguji sampel dari pola karakter yang digambar pada area kanvas lakukan klik pada tombol kenali karakter. *Dialogbox* keterangan hasil pengenalan akan menyatakan apakah pola karakter yang digambar sesuai dengan karakter yang telah disimpan.

5.1.4. Impelementasi *Form About&Form Author*

Gambar 5.6. ***Form About***

Gambar 5.7. ***Form Author***

Form about & form author berisi penjelasan mengenai aplikasi dan cara penggunaannya secara singkat.

5.2. Pengujian

Pengujian dilakukan untuk mengetahui seberapa tinggi tingkat akurasi pengenalan pola yang dilakukan oleh metode Kohonen *neural network* dan melihat apakah aplikasi yang dibuat sesuai dengan analisa dan perancangan. Pengujian dilakukan dengan membandingkan beberapa pola *sample* untuk tiap huruf arab yang digambar pada kanvas dengan pola karakter yang telah disimpan sebelumnya.

5.2.1. Pengujian *Black Box*

Pengujian *black box* digunakan untuk menguji komponen-komponen yang ada pada aplikasi tanpa memperhatikan struktur logika internalnya.

Tabel 5.1. Pengujian *Black Box*

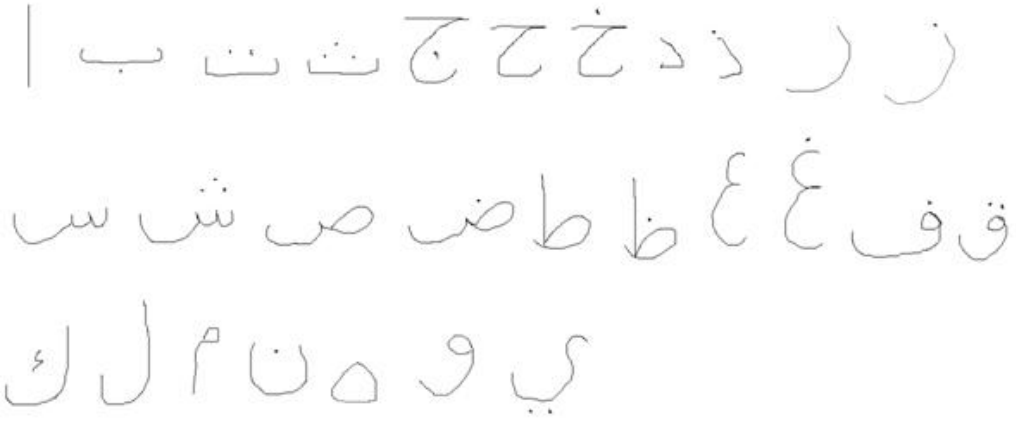
No	Komponen yang diuji	Properti Komponen	Hasil Yang Diharapkan	Hasil Pengujian
1	*Main Form*	Tombol menu *input character*	Menampilkan *form input*	Benar
		Tombol menu *recognize character*	Menampilkan *form recognize character*	Benar
		Tombol menu	Menampilkan *form about*	Benar

		About		
		Tombol Menu *Author*	Menampilkan *form author*	Benar
2	*Form Input Character*	Tombol muat karakter	Memuat *list* karakter	Benar
		Tombol hapus karakter	Menghapus karakter dari *list* karakter	Benar
		Tombol simpan	Menyimpan karakter yang telah ditambahkan pada *list*	Benar
		Tombol tambah karakter	Menambahkan pola karakter kedalam *list*	Benar
		Tombol karakter berikutnya	Melakukan seleksi pada karakter pada *list* karakter	Benar
		Tombol *clear*	Membersihkan area gambar atau kanvas	Benar
3	*Form Recognize Character*	Tombol muat karakter	Memuat *list* karakter	Benar
		Tombol *Begin Training*	Melakukan *training* karakter yang ada pada *list*	Benar
		Tombol karakter berikutnya	Melakukan seleksi pada karakter pada *list* karakter	Benar
		Tombol *clear*	Membersihkan area gambar atau kanvas	Benar

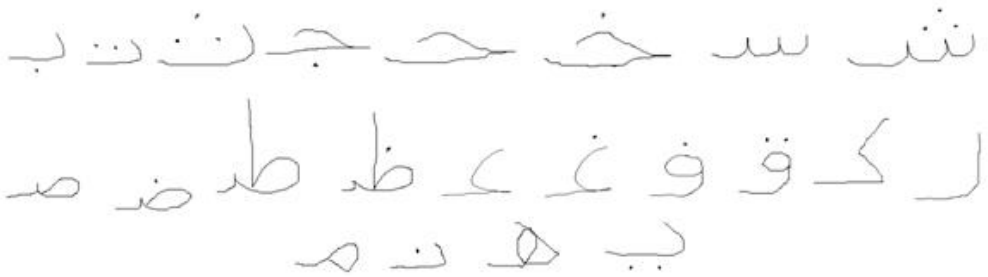
		Tombol konversi biner	Menampilkan jendela yang berisi representasi biner dari karakter	Benar
4	*Form About*		menampilkan tulisan mengenai aplikasi	Benar
5	*Form Author*		menampilkan biodata pengembang aplikasi	Benar

5.2.2. Pola Karakter yang Diuji

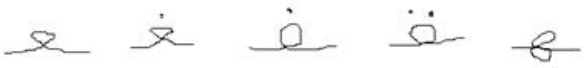
Berikut adalah pola-pola karakter huruf arab yang dijadikan sampel pengujian yang terdiri dari *isolated* (terpisah), *beginning* (awal), dan sebagian kecil *middle* (tengah) dan *end* (akhir).



Gambar 5.8. Sampel Pola Karakter *Isolated* (Terpisah)



Gambar 5.9. Sampel Pola Karakter *Beginning* (Awal)



Gambar 5.10. Sampel Pola Karakter *Middle* (Tengah)

Gambar 5.11. Sampel Pola Karakter *End* (Akhir)

5.2.3. Hasil Pengujian

Tahapan pengujian dilakukan dengan beberapa kriteria sebagai berikut :

1. Pengujian dibagi kedalam tiga bagian jumlah sampel, 3 sampel karakter *isolated* yang berjumlah 15 pola karakter pengujian, seluruh karakter *isolated* yang berjumlah 140 pola karakter pengujian dan gabungan semua karakter *isolated*, *beginning*, *middle*, dan *end* yang berjumlah 310 pola karakter pengujian.
2. Tiap karakter diberikan 5 pola sampel yang berbeda.

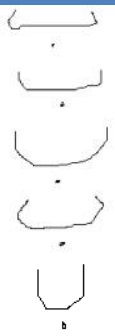
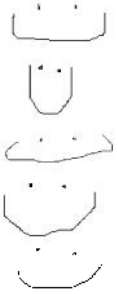
3. Persentase pengenalan dibagi kedalam tiga kategori banyaknya jumlah sampel seperti yang dijelaskan pada no.1. Jumlah persentase ditentukan oleh banyaknya karakter yang dikenali.

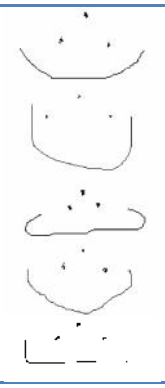
4. Tingkat pengenalan karakter dikatakan sangat baik apabila tingkat akurasinya mencapai lebih dari 90% *excellent* (tinggi sekali), 89%-80% *very good* (sangat tinggi), 79%-64% *satisfactory* (tinggi), 63%-51% *sufficient* (cukup), 50%-0% *insufficient* (rendah).

Untuk beberapa hasil pengujian pengenalan karakter huruf arab dapat dilihat seperti berikut (selengkapnya pada lampiran) :

1. Pengujian dengan 3 sampel karakter huruf arab *isolated* (terpisah)

Tabel 5.2. Pengujian dengan sampel 3 karakter huruf arab *isolated* (terpisah)

Karakter	Pola	Hasil Pengenalan	Persentase *Error*
ب		ب ب ب ب ب	0/5 x 100% = 0%
ت		ت ت ت ت ت	0/5 x 100% = 0%

ش		ت ت ش ت ش	3/5 x 100% = 60%

Dari hasil pengujian tahap pertama yang menggunakan tiga sampel pola karakter huruf arab *isolated* didapatkan rata-rata total *error* seperti berikut :

Rata-rata *error* = jumlah *error* / jumlah sampel uji x 100%

= 3/15 * 100% = 20%

Persentase pengenalan karakter = 100% - 20% = 80%

2. Pengujian dengan semua sampel karakter huruf arab *isolated* (terpisah)

Tabel 5.3. Pengujian dengan sampel semua karakter huruf arab *isolated* (terpisah)

Karakter	Pola	Hasil Pengenalan	Persentase *Error*
ا		ا ج ل ر د	4/5 x 100% = 80%

	ا		
ب		ب ب ب ب ب	0/5 x 100% = 0%
ت		ت ت ت ت ت	0/5 x 100% = 0%
ث		ت ت ث ت ث	3/5 x 100% = 60%
ج		ن ج ح ه ج	3/5 x 100% = 60%

Untuk keseluruhan hasil pengujian dapat dilihat pada lampiran A. Total *error* yang ada pada pengujian dengan keseluruhan sampel karakter *isolated* yang berjumlah 28 adalah :

Rata-rata *error* = jumlah *error* / jumlah sampel uji x 100%

= 32/140 * 100% = 22,86%

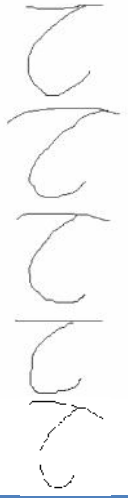
Persentase pengenalan karakter = 100% - 22,86% = 77,14%

3. Pengujian dengan gabungan sampel karakter huruf arab *isolated* (terpisah), *beginning* (awal), *middle* (tengah), *end* (akhir).

Tabel 5.4. Pengujian dengan sampel karakter huruf arab *isolated* (terpisah), *beginning* (awal), *middle* (tengah) dan *end* (akhir)

Karakter	Pola	Hasil Pengenalan	Persentase *Error*
		ا ج ل ر د	4/5 x 100% = 80%

ب		ب ب ب ب ب	0/5 x 100% = 0%
ت		ت ت ت ت ت	0/5 x 100% = 0%
ث		ت ت ث ت ث	3/5 x 100% = 60%
ج		ن ج ح ه ج	3/5 x 100% = 60%

ح		ح ح ح ح ح م	1/5 x 100% = 20%

Untuk keseluruhan hasil pengujian dapat dilihat pada lampiran B. Total *error* yang ada pada pengujian dengan keseluruhan sampel karakter *isolated, beginning,* dan beberapa *middle* dan *end* yang berjumlah 62 karakter adalah sebagai berikut :

Rata-rata *error* = jumlah *error* / jumlah sampel uji x 100%

= 92/310 * 100% = 29,68%

Persentase Pengenalan karakter = 100% - 29,68%

= 70,32%

Tabel 5.5. Hasil Pengujian Sampel

Jumlah Sampel	**Persentase *Error***
15	20%
140	22,86%
310	29,68%

Dari tabel 5.14. dapat dilihat persentase *error*hasil pengujian terhadap sampelpola karakter huruf arab yang ketika karakter yang di *training* dan jumlah pola sampel semakin banyak maka akan mempengaruhi proses pengenalan yang dilakukan

oleh algoritma Kohonen dikarenakan pola karakter yang dianggap mirip karakter yang lain.

Tabel 5.6. Hasil Pengujian Berdasarkan Kategori Huruf

Kategori Huruf	Jumlah Karakter yang Dikenali	Total Karakter yang Diuji	Persentase *Error*
Tunggal (*Isolated*)	85	110	22,73%
Awal (*Beginning*)	62	110	43,64%
Tengah (*middle*)	19	25	24%
Akhir (*end*)	32	35	8,57%

Tabel diatas menunjukkan persentase *error*yang diambil berdasarkan kategori huruf arab dengan total sampel pengujian 310 karakter. Dari hasil yang ditampilkan dapat dilihat karakter huruf arab tunggal (*isolated*) mempunyai persentase *error*22,73%, huruf arab kategori diawal kata (*beginning*) 43,64%, huruf arab ditengah kata (*middle*) 24% dan diakhir kata (*end*) 8,57%.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan pengujian yang dilakukan dengan menggunakan metode Kohonen *neural network,* didapat kesimpulan sebagai berikut :

1. Metode Kohonen *neural network* dapat melakukan proses pengenalan karakter huruf arab dengan kategori tingkatan tinggi (*satisfactory*) dan sangat tinggi (*very good*).
2. Persentase *error* Kohonen *neural network* dalam proses pengenalan karakter mencapai 80% (*very good*) untuk 15 sampel pola yang terdiri dari 3 huruf *isolated* (terpisah), 77,14% (*satisfactory*) untuk 140 sampel pola yang terdiri dari keseluruhan bentuk huruf *isolated* (terpisah) dan 70,32% (*satisfactory*) untuk 310 sampel pola yang terdiri dari keseluruhan bentuk huruf *isolated* (terpisah), dan beberapa dari bentuk *beginning* (awal), *middle* (tengah), serta *end* (akhir).
3. Semakin tingginya tingkat *error* ketika sampel pola ditambah adalah dikarenakan banyaknya pola karakter huruf arab yang duji memiliki kemiripan bentuk antara satu dengan lainnya, serta dikarenakan bentuk karakter yang lebih rumit dibanding karakter-karakter huruf lain.

6.2. Saran

Ada beberapa hal yang dapat dijadikan pertimbangan untuk perbaikan penelitian menggunakan metode Kohonen *neural network* ini kedepannya, antara lain:

1. Untuk mempermudah proses pengenalan dan pengujian karakter, proses *input* karakter yang saat ini dengan cara menggambarkan pola karakternya pada area kanvas dapat diganti dengan cara membangun aplikasi pengenalan

karakter dengan menggunakan objek gambar sebagai proses *input* karakter yang akan dikenali dan diuji.

2. Aplikasi pengenalan karakter dengan menggunakan metode Kohonen *neural network ini* dapat dikembangkan untuk dapat mengenali suku kata dan kata-kata dari huruf arab.

DAFTAR PUSTAKA

Al Jawfi, Rashad,"*Handwriting Arabic Character Recognition LeNet Using Neural Network"*, 2007.

Asworo, "*Perbandingan Antara Metode Kohonen Neural Network dan Learning Vector Quantization Pada Sistem Pengenalan Tulisan Tangan Secara Real Time"*, 2010.

Athimethphat, Maythapolnun, "*A Review on Global Binarization Alghorithms for Degraded Document Images*", 2011

Bellal, A.K.M Bellal,"*Hand Written Character Recognition System Using Kohonen Self Organization Map"*, 2010.

Black, Rex, "*Pragmatic Software Testing : Becoming an Effective and Efficient Test Professional.*", Wiley Publishing Inc, Hoboken, 2007.

Daya, Peter, "*Unsupervised Learning*", Wilson, RA & Keil, F, editors, The MIT Encyclopedia Of The Cognitive Sciences, 1999.

Heaton, Jeff. "*Introduction to Neural Networks for Java"*, Heaton Research, 2005.

Hartanto, Suryo, "*Optical Character Recognition Menggunakan Algoritma Template Matching Correlation*", Journal of Informatics and Technology, Vol 1, 2012.

Iskandar, Reinaldy,"*Perancangan Program Aplikasi Pengenalan Pola Abjad Arab Menggunakan Metode transformasi Wavelet Dan Back Propagation"*, 2011.

Koerich, Alessandro L, "*Unconstrained Handwritten Character Recognition Using Different Classification Strategies*", 2003.

Lorigo, Liana M,"*Off-line Arabic Handwritten Recognition : A Survey"*, IEEE Transaction On Pattern Analysis And Machine Intelligence, 2006.

Mezghani, Neila, "*Online Persian/Arabic Character Recognition by Polynomial Representation and a Kohonen Neural Network"*, The 18th International Conference on Pattern Recognition, 2006

Mubarok, "*Pengenalan Tulisan Tangan Aksara Sunda Menggunakan Kohonen Neural Network"*, 2011.

Munawari, Akhmad, "*Belajar Cepat Tata Bahasa Arab Program 30 Jam : Nahwu Shorof Sistematis*", 2006.

Nourouzian, Ehsan, "*Online Persian/Arabic Character Recognition by Polinomial Representation And Kohonen Network"*, 2010.

Otsu, Nobuyuki, "*A Threshold Selection Method from Gray-Level Histogram"*, IEEE Transaction On Systems, Man, And Cybernetics, Vol.SMC-9, No.1 Januari 1979.

Putra, Dharma, "*Pengolahan Citra Digital"*, Penerbit Andi, Yogyakarta, 2010.

Peyarajan, S, "*On-Line Tamil Hand Written Character Recognition Using Kohonen Neural Network"*, Research Journal of Computer Systems Engineering-An International Journal, 2 Juli 2011.

Ratih, Eca Indika, "*Aplikasi Jaringan Syaraf Tiruan Untuk Pengenalan Karakter Dengan Metode Hopfield*", 2009.

Sankur, Bulent, "*Survey Over Image Thresholding Techniques And Quantitative Performance Evaluation*", 2004.

Santosa, Budi, "*Data Mining : Teknik Pemanfaatan Data Untuk Keperluan Bisnis"*, Penerbit Graha Ilmu, Yogyakarta, 2007.

Setiawan, Arif, "*Analisa Sistem Pengenalan Karakter Menggunakan Jaringan Syaraf Tiruan Untuk Pembacaan Dokumen Yang Rusak Karena Banjir*", 2010.

Shatil, Adnan Md. Shoeb,*"Bangla Optical Character Recognition Using Kohonen Network"*, 2006.

Subiyanto, "*Sistem Komputasi Jaringan Syaraf Tiruan Backpropagation*", 2010.

Syamani, "*Komparasi Algoritma Non-Parametik k-Nearest Neighbour Classifier Menggunakan Euclidean Distance dan Manhanttan Distance Untuk Klasifikasi Multispektral Tutupan Lahan*", PIT MAPIN XVII, Bandung, 10 Desember 2008.

Tae, Gadis Fransiska Yulianti, "*Penerapan Kohonen Self Organized Map Dalam Kuantisasi Vektor Pada Kompresi Citra Bitmap 24 Bit*", 2010.

Tian, J, "*Software Quality Engineering : Testing, Quality Assurance, and Quantifiable Improvement*", John Wiley & Sons, Inc, Hoboken, 2005.

Ummami, Muhammad Amrul, *"Analisa dan Implementasi Pengenalan Huruf Arab Menggunakan Modified Direction Feature Extraction dan Learning Vector Quantization"*, 2010.